UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Desarrollo de una aplicación web para la gestión de comunidades de vecinos

Silvia Nerea Anguita de Blas
Tutor: Daniel Hernández Lobato

JUNIO 2016

Desarrollo de una aplicación web para la gestión de comunidades de vecinos

AUTOR: Silvia Nerea Anguita de Blas
TUTOR: Daniel Hernández Lobato

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2016

Resumen (castellano)

La gestión de comunidades de vecinos suele convertirse en un trabajo pesado y tedioso debido a la falta de comunicación entre los vecinos, ya sea por no disponer del tiempo necesario o por falta de interés. Este problema se puede solucionar llevando las comunidades de vecinos a la web, como ya han hecho varias empresas. Sin embargo, estas empresas se basan en la gestión de comunidades de vecinos a través un profesional. El cuál es el que decide en la mayoría de los casos que información es la que podrán visualizar los usuarios.

La falta de autogestión en las webs disponibles en el mercado no es el único problema. Las aplicaciones disponibles muestran interfaces con demasiada información haciendo que la navegación sea una barrera para aquellas personas con medios o bajos conocimientos sobre la navegación web, teniendo en cuenta que la mayoría de propietarios de viviendas son personas de mediana y tercera edad.

El objetivo principal de este Trabajo de Fin de Grado es el desarrollo de Community, una aplicación web orientada a la autogestión de comunidades de vecinos. De este modo, se proporciona al usuario una herramienta con la cual mantenerse informado y conectado con su comunidad de vecinos.

Las ventajas de Community sobre otros sistemas con los mismos objetivos, es que los usuarios podrán observar toda la información sin limitaciones y dispondrán de una interfaz sencilla e intuitiva facilitando su uso a aquellas personas que no dispongan de conocimientos avanzados sobre navegación web.

El desarrollo de esta aplicación se ha realizado por medio de JSF, que es la tecnología recomendada en la actualidad para el desarrollo web de aplicaciones basadas en Java. Junto con Primefaces, para proporcionar a la aplicación componentes complejos, pero de uso sencillo, y la base de datos PostgreSQL.

Palabras clave (castellano)

Comunidad de vecinos, aplicación web, JSF, Primefaces y PostgreSQL.

Abstract (English)

The management of neighborhood communities often becomes a heavy and tedious work due to the lack of communication among neighbors, that either do not have the time or the interest. This problem can be solved by bringing neighborhood communities to the web, as several companies have done. However, these companies are based on the management of neighborhood communities through a professional. Which it is the one who decides in most cases what information the users could view.

The lack of self-management in the websites available is not the only problem. The interfaces of the available applications show too much information making navigation a barrier for those with medium or low knowledge about web browsing, considering that most homeowners are middle-aged and elderly.

The main objective of this Bachelor Thesis is the development of Community, a web application oriented to self-management of neighborhood communities. Thereby, the user is provided with a tool which makes them stay informed and connected to their community of neighbors.

The advantages of Community over other systems with the same objectives, is that users will see all the information without limitations and have a simple and intuitive interface facilitating its use to people who do not have advanced knowledge of web browsing.

The development of this application has been made with JSF, which is currently the recommendation for the development of applications based on Java technology. Along with Primefaces, to provide complex components but with simple use, and PostgreSQL database.

Keywords (inglés)

Neighborhood communities, web application, JSF, Primefaces and PostgreSQL.

Agradecimientos

Primero quiero agradecer a mi familia por todo el apoyo que me ha brindado estos últimos cuatro años. Gracias por dejarme elegir mi camino, por ayudarme siempre en todo lo posible y por estar siempre a mi lado. Muchas gracias papá, por mostrarme lo que es el trabajo duro y por ser mi ejemplo a seguir.

Muchas gracias, a esas personas maravillosas que puedo llamar amigos y que si no fuese por la universidad no hubiese conocido. Gracias Oscar, Ángel y Alejandro. Y muchas gracias Patri, desde el primer hasta el último día. Gracias por todos los momentos que hemos pasado juntos, os quiero.

Quiero agradecer a mi tutor, Daniel, por haber estado atento en todo momento y por haberme ayudado en todo lo posible.

Por último, muchas gracias David. Gracias por aguantarme siempre y por mostrarme que puedo conseguir todo lo que quiera. Gracias por hacerme feliz. Por nosotros.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

1 Introducción

1.1 Motivación

Hoy en día el problema más recurrente en las comunidades de vecinos suele ser la falta de comunicación e información, ya sea por falta de tiempo o de interés. “Los españoles no suelen ser muy participativos en los asuntos de su comunidad porque huyen de los conflictos inherentes. El uso de las nuevas tecnologías es una buena forma de intervenir sin verse afectado por los problemas asociados a las reuniones de vecinos”, Salvador Díez, presidente del Consejo General de Colegios de Administradores de Fincas de España (CGCAFE). [1]

Actualmente, existen aplicaciones web que llevan a cabo esta función de unir a las personas de una misma comunidad. Muchas de ellas están orientadas a la gestión administrativa de las comunidades y a su uso por parte de profesionales. Como consecuencia, la interacción de los usuarios con la web no suele presentar buenos resultados.

La motivación de este proyecto es el desarrollo de una aplicación que una a personas que viven en una misma comunidad de vecinos, ofreciendo prestaciones que fomenten la comunicación y faciliten el trabajo administrativo y de gestión, pero de manera que no haya usuarios discriminados por sus conocimientos tecnológicos.

1.2 Objetivos

El objetivo de este Trabajo de Fin de Grado es el diseño y desarrollo de Community, una aplicación web que permite la comunicación y autogestión de comunidades de vecinos por parte de los integrantes de estas.

Las principales funcionalidades que debe presentar la aplicación son:

- **Gestión de incidencias:** Permitiendo al usuario visualizar, crear, editar y borrar incidencias. Solo podrán realizar la edición y la eliminación el autor o el presidente de la comunidad.

- **Gestión de un tablón de anuncios:** Permitiendo al usuario visualizar, crear, editar y borrar anuncios. Solo podrán realizar la edición y la eliminación el autor o el presidente de la comunidad.

- **Gestión de usuarios:** Los usuarios podrán ver una lista de los demás usuarios que pertenecen a su comunidad y visualizar la información de contacto de estos.

- **Gestión de reuniones:** La aplicación permitirá al presidente crear, editar y borrar reuniones de un calendario y a los demás usuarios les permitirá visualizarlas.

- **Gestión de gastos:** La aplicación permitirá al presidente crear, editar y borrar gastos y a los demás usuarios les permitirá visualizarlos tanto en formato de gráfica como en formato de tabla.

- **Gestión de votaciones:** La aplicación permitirá al presidente crear, editar, borrar votaciones y ver los resultados en forma de gráfica y los vecinos de la comunidad podrán votar.

- **Módulo de mensajería:** La aplicación dispondrá de un módulo de mensajería privada en el cual los usuarios podrán enviar mensajes a otros vecinos de su comunidad, y ver su bandeja de entrada y de salida.

- **Comprobación de Comunidad:** La aplicación dispondrá de una sección donde los usuarios podrán comprobar si su comunidad se encuentra registrada.

- **Registro:** Si la comunidad se encuentra registrada el usuario podrá registrarse en la aplicación como vecino de esta. Si por el contrario la comunidad no está registrada el usuario podrá registrarse y se convertirá en el presidente virtual de su comunidad.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- En el capítulo 2, Estado del arte, se han realizado dos estudios. El primero es para saber qué es lo que podemos encontrar en el mercado relacionado con la gestión de comunidades de vecinos y de qué forma se puede mejorar. En el segundo estudio se investigan las tecnologías destacadas para el desarrollo de aplicaciones web y una vez seleccionada la más adecuada, se especifican las demás tecnologías que serán empleadas.

- En el capítulo 3 se abarca el diseño de la aplicación incluyendo diagramas de casos de uso y de secuencia, el diseño de la base de datos y se especifica el ciclo de vida que ha seguido el proyecto.

- En el capítulo 4 describe el modelo, las capas del proyecto y la forma en la que las diferentes tecnologías han sido empleadas.

- En el capítulo 5 se detallan las pruebas realizadas.

- En el capítulo 6 se presentan las conclusiones tras la finalización de la primera versión del proyecto, y se plantean posibles ampliaciones y mejoras para la aplicación web.

2 Estado del arte

2.1 Análisis de sistemas de gestión de comunidades de vecinos

Antes de comenzar con el desarrollo de la aplicación se realizó un estudio sobre aplicaciones web ya existentes orientadas a la gestión de comunidades de vecinos. Durante este estudio se observó que existen multitud de aplicaciones con la misma orientación, algunas más dirigidas a profesionales de la gestión y otras por el contrario dirigidas a usuarios sin muchos conocimientos en cuanto a la gestión de comunidades se refiere. La finalidad de este estudio era la recolección de información sobre los principales sistemas en este campo y así poder analizar los puntos fuertes y débiles de estos.

- **Vecinos.net** (https://vecinos.net/)

Figura 2-1: Logo Vecinos.net

Es una de las aplicaciones web más destacadas en este campo. Según enuncian en su web, facilita la comunicación entre los miembros de la comunidad vía online y permite una gestión más clara al disponer de un medio donde poder intercambiar y debatir asuntos de interés.

Vecinos.net ha sido una de las aplicaciones web seleccionadas para su estudio por su semejanza en los objetivos principales con nuestra aplicación web.

Esta web tiene dos modos de uso: modo vecino y modo administrador. El modo administrador está dirigido para profesionales que podrán usar la web como herramienta de trabajo. Para ser empleada de este modo el administrador debe pagar una tarifa. Este pago incluye la instalación del dominio y la marca, una web corporativa y las opciones de gestión avanzada de las comunidades. El administrador será el encargado de crear las comunidades en su web y los vecinos accederán al dominio del administrador para registrarse. Este modo dispone de un panel de administración donde se muestran todas las comunidades que gestiona, tiene la posibilidad de personalizar la apariencia de la web y puede ver estadísticas de uso de cada una de las comunidades que gestiona. Este modo pone a disposición del administrador herramientas avanzadas para la gestión de la contabilidad.

Las funcionalidades principales que comparten ambos modos son: tablón de comunicados, gestión de incidencias, documentos compartidos, sistema de votaciones, sistema de mensajería, galería de fotos y listado de vecinos.

El modo vecino se emplea cuando la comunidad no dispone de administrador y se lleva a cabo una autogestión. Dentro de este modo hay dos tarifas una gratuita, con 100MB de almacenamiento y con publicidad emergente, y otra Premium, la cual tiene un coste de 100€ por año, el almacenamiento aumenta a

los 1000MB y no se muestra publicidad. La tarifa Premium además de disponer de todas las funcionalidades principales cuenta con una sección gestionar la reserva de las instalaciones, una sección de contratos con profesionales y proveedores de servicios. El modo vecino no dispone de ningún módulo para gestionar la contabilidad.

Tras utilizar la versión demo que se encuentra en la web he podido observar que a simple vista dispone de una interfaz muy clara y sencilla pero a la hora de utilizarla como un vecino se vuelve demasiado complicada y la interacción no es intuitiva lo que puede suponer un gran inconveniente para un usuario con bajos e incluso medios conocimientos de navegación web. La publicidad que aparece en la versión gratuita es demasiado grande y aparece en medio de los contenidos de la web.

La web dispone de una opción de búsqueda de gestores de comunidades cercanos que utilicen esta herramienta.

- **VecinosEnRed** (http://www.vecinosenred.es/)

Figura 2-2: Logo VecinosEnRed

VecinosEnRed es un software para la administración de la comunidad de propietarios que facilita la gestión de las comunidades de vecinos y su mantenimiento [2]. Es una aplicación que permite gestionar todas las tareas administrativas asociadas a una comunidad, y compartir de manera interactiva la información con los propietarios.

Entre las principales tareas que se pueden realizar en esta web se encuentran la gestión de presupuestos, control de ingresos y gastos, informes de estado financiero, control de morosos, seguimiento de incidencias, administración de reservas de los espacios comunes y comunicación entre los propietarios.

Al igual que la web anterior dispone de dos modos de empleo: modo administrador y modo vecino.

El modo administrador dispone de multitud de módulos: un portal del empleado para visualizar las novedades de cada comunidad, módulo de gestión y reserva de instalación desde donde se podrá gestionar el pago del alquiler de las instalaciones donde sea necesario, módulo presupuestario para crear los presupuestos y reflejar los ingresos y los gastos, un panel de estado de la

comunidad donde se visualiza información de manera comparativa entre los datos presupuestados y la situación real y un panel de estado financiero.

El modo vecino permite crear incidencias, reservar áreas comunes, enviar mensajes al administrador, ver el estado de las cuentas, votar en encuestas y revisar el calendario de eventos.

VecinosEnRed dispone de tres tarifas las cuales ofrecen las mismas prestaciones, pero se diferencian en el número máximo de vecinos que puede haber: Free en la cual puede haber un máximo de 20 vecinos en la comunidad, Medium con una cuota de 90€ en la cual puede haber un máximo de 100 vecinos y Enterprise con una cuota de 190€ en la cual no hay limitación en el número de vecinos. Todas las cuotas ofrecen acceso a 5 vecinos a la web y si se quiere aumentar este número hay que realizar un pago anual adicional, lo que al final supone un coste muy elevado si se desea que todos los vecinos dispongan de acceso.

A diferencia de la web anterior, VecinosEnRed está más orientada al campo administrativo, con muchas más prestaciones en este módulo. Esto se puede ver como una ventaja si el administrador de la comunidad es un profesional del campo, pero si por el contrario la comunidad se encarga de autogestionarse y el administrador es un vecino este módulo puede ser demasiado complejo. La navegación por la página es mucho más complicada comparada con la web anterior, dado que cada pantalla se encuentra muy sobrecargada mostrando demasiado información.

2.1.1 Conclusión sobre el estudio de aplicaciones similares

Como se ha podido observar en el apartado anterior, las aplicaciones sobre las que se ha realizado el estudio muestran objetivos similares a los presentados para el proyecto. Todas las aplicaciones web estudiadas se caracterizan por su orientación a la gestión de comunidades de vecinos en las cuales el mayor inconveniente que presentan es lo complicada que resulta la navegación. Esto puede ser consecuencia de querer ofrecer un gran número de servicios avanzados, marginando inconscientemente a los usuarios con bajos o medios niveles de navegación web.

La mayoría de las aplicaciones de este tipo están más orientadas a su uso como herramienta de comunicación entre el administrador (siendo este un profesional) y los vecinos, más que una herramienta para que los propios vecinos puedan autogestionarse. Otro inconveniente que presentan estas páginas es que la figura del administrador es la encargada de indicar que información es visible para los usuarios.

A partir del estudio se ha podido determinar los puntos a mejorar con respecto a éstas, destacando una interfaz más simple que permita una navegación más sencilla e intuitiva para que de este modo sea apta para todos los usuarios y prestaciones que permitan gestionar la administración de manera sencilla para que puedan ser empleadas por usuarios no profesionales de la administración.

2.2 Estudio de tecnologías de desarrollo web

En este apartado se realiza un análisis de las principales tecnologías de desarrollo web disponibles en el mercado para la posterior selección de la tecnología que más se adecue y más beneficiosa resulte para la implementación de los requisitos del proyecto.

Las tecnologías líderes en el desarrollo de aplicaciones Web son principalmente ASP.NET, PHP y JSP [3].

- **ASP.NET**. Es un framework para aplicaciones web desarrollado y comercializado por Microsoft [4].

  Entre las ventajas que podemos encontrar es que al formar parte del framework .NET se puede elegir cualquier lenguaje soportado por el marco de trabajo de este para desarrollar la aplicación web, el cual soporta ya más de 20 lenguajes y los lenguajes son orientados a objetos. Dispone de la biblioteca FCL (Framework Class Library) la cual proporciona un alcance comparable a las bibliotecas estándar de Java y permite el uso de XML, operaciones de lectura y escritura de archivos, acceso a datos, etc., detecta el tipo de navegador utilizado por el cliente a la hora de realizar una petición al servidor y, en consecuencia, determina la versión de HTML que éste soporta.

  Como desventajas de esta tecnología destacamos que depende de sistemas Microsoft, es una plataforma de pago y tiene un gran consumo de recursos lo, que requiere servidores de mayor capacidad o mayor número de servidores.

- **PHP.** Es considerado uno de los lenguajes más flexibles, potentes y de alto rendimiento conocidos hasta día de hoy. Algunos de los puntos fuertes de esta tecnología es que presenta una curva de aprendizaje muy baja, lo que permite a la mayoría de los programadores crear aplicaciones complejas y es un lenguaje multiplataforma [5].

  Por el contrario, algunos de sus puntos débiles son, por ejemplo, que no ofrece las mismas posibilidades que otras tecnologías en lo referido a modularización y organización en capas, el acceso a base de datos no está estandarizado y según estudios no es la tecnología más adecuada para aplicaciones de grandes dimensiones.

- **JavaServer Pages (JSP).** Es una tecnología que permite el desarrollo de páginas web dinámicas, de una forma simplificada y rápida, basadas principalmente en HTML y XML, entre otros, y se distingue de otras tecnologías por su uso de Java como lenguaje de programación [6].

  Como ventajas destacamos que java es un lenguaje orientado a objetos y JSP dispone a su alcance de todas las ventajas que ofrece Java. JSP es multiplataforma, posee a su disposición multitud de frameworks que agilizan y facilitan la implementación de aplicaciones web como Spring, Tapestry, Hibernate, Struts, etc. y se construye sobre la API de Java Servlets lo que permite el acceso a la API Java Enterprise incluyendo: API JDBC que permite el soporte de acceso a base de datos, EJB, etc.

Como desventajas destacamos que la curva de aprendizaje es mayor en relación con otras tecnologías, como por ejemplo PHP, no resulta de gran utilidad para proyectos pequeños debido a que conlleva más tiempo de desarrollo y puede no existir JDBC para bases de datos poco comerciales.

A pesar de que las tecnologías anteriores son las más utilizadas para el desarrollo web, existen tecnologías más modernas que cada vez se emplean más, entre las que destacan: Ruby on Rails, Django y JSF.

- **Ruby on Rails (RoR).** Es un framework de desarrollo web de código abierto escrito en el lenguaje Ruby, está optimizado para la satisfacción de programadores y para la productividad sostenible y sigue el modelo de arquitectura MVC [7]. Se basa en dos principios: DRY (Don't Repeat Yourself), sugiere la eliminación de código repetido y "Convención sobre configuración" cuyo significado es que Rails hace suposiciones sobre lo que el programador quiere hacer y cómo lo va a hacer en lugar de requerir especificaciones sobre todas las configuraciones [8].

  Entre los puntos fuertes podemos destacar que Ruby es un lenguaje orientado a objetos, como ya hemos visto uno de los principios de RoR es que favorece el ahorro de líneas de código, ofrece un conjunto extenso de librerías que facilitan el desarrollo web y al ser una de las principales tecnologías empleadas en la actualidad para el desarrollo web cuenta con una extensa comunidad detrás que la respalda.

  Como puntos débiles el programador debe tener conocimientos sobre Ruby y puede resultar complicado encontrar la documentación adecuada para determinadas gemas y librerías.

- **Django.** Es un framework de desarrollo web de alto nivel que hace uso del lenguaje Python, y según enuncian en su web ahorra tiempo y hace que el desarrollo web sea más dinámico [9].

  Entre los puntos fuertes de esta tecnología destacamos que dispone de una extensa cantidad de librerías que permiten el desarrollo sin necesidad de escribir mucho código y que al ser una de las principales tecnologías empleadas en la actualidad cuenta con una extensa comunidad detrás que la respalda.

  Como puntos débiles el programador debe tener conocimientos sobre Python y cuenta con menos documentación que otras tecnologías.

- **JavaServer Faces (JSF).** Es la sucesión de JSP, es un framework para aplicaciones Java basadas en Web cuyo rasgo principal es la simplificación en el desarrollo de interfaces de usuario y en estos momentos es la tecnología recomendada para la construcción de aplicaciones web basadas en Java [10].

  Esta tecnología incluye un gran conjunto de etiquetas de alto nivel mediante las cuales se simplifica el desarrollo de las interfaces de usuario. Otra gran ventaja es que las últimas versiones de este framework disponen de soporte implícito para AJAX lo que hace que el uso de esta tecnología sea más sencillo, sin

necesidad de usar código en Javascript y permite la exportación del modelo de manera sencilla mediante servicios web.

2.2.1 Conclusión sobre el estudio de tecnologías de desarrollo web

Llegados a este punto, tras el estudio de las tecnologías mencionadas, se tomó la decisión de que la tecnología que más se adecúa a las necesidades del proyecto es JSF. Entre las razones de esta decisión destaca: la amplia documentación existente y la gran comunidad de desarrolladores que presenta esta tecnología; el conocimiento sobre Java por parte de la desarrolladora; y el uso de librerías y etiquetas de alto nivel que facilitan el desarrollo.

2.3 Otras tecnologías empleadas

- **Primefaces.** Es una librería de componentes de alto nivel para interfaces gráficas basadas en JSF. El conjunto de componentes que ofrece facilitan en gran medida la creación de interfaces de usuario para aplicaciones web [11].

  Como puntos fuertes de esta tecnología destacamos que es fácil de trabajar y es muy ligera. Dispone de una gran selección de componentes y estos tienen un diseño sencillo e innovador haciendo muy intuitivo su uso para los usuarios finales. Dispone de un kit para crear aplicaciones web para dispositivos móviles, 38 temas gratuitos de apariencia para los componentes y ofrece soporte nativo de Ajax, el cual permite al desarrollador indicar que componentes de la página serán los que se actualizarán.

  A pesar de ser una tecnología relativamente nueva dispone de muy buena y extensa documentación y una comunidad muy activa de desarrolladores. En este proyecto se ha empleado la versión 5.3.

Figura 2-3: Logo Primefaces

- **JDBC.** Se ha hecho uso de esta API para realizar la interacción entre la aplicación y la base de datos, dado que nos permite establecer la conexión con la base de datos y la ejecución de operaciones sobre esta desde Java mediante sentencias SQL [12].

- **PostgreSQL y PgAdmin3.** La definición de la base de datos se ha realizado mediante el lenguaje SQL a través del sistema de gestión de bases de datos relacionales PostgreSQL, en la versión 9.3.10 [13].

  Para facilitar la gestión y administración de la base de datos se ha utilizado el software PgAdmin3. El cual es una herramienta de administración de bases de datos, que nos permite acceder a todas las funcionalidades de esta.

- **CSS3.** Es el último estándar de CSS. Este lenguaje ha sido empleado para definir la presentación de los documentos XHTML y para modificar la apariencia de los componentes de Primefaces [14].

- **AJAX.** Es una técnica de desarrollo web para crear aplicaciones interactivas [15]. Añade dinamismo a las páginas web. El navegador interactúa con el servidor sin recargar la página web, para ello se usan llamadas asíncronas (no bloqueantes).

3 Diseño

3.1 Ciclo de vida

Para el desarrollo del proyecto se ha seguido el ciclo de vida basado en un modelo cascada con retroalimentación, como el que se muestra en la siguiente figura:

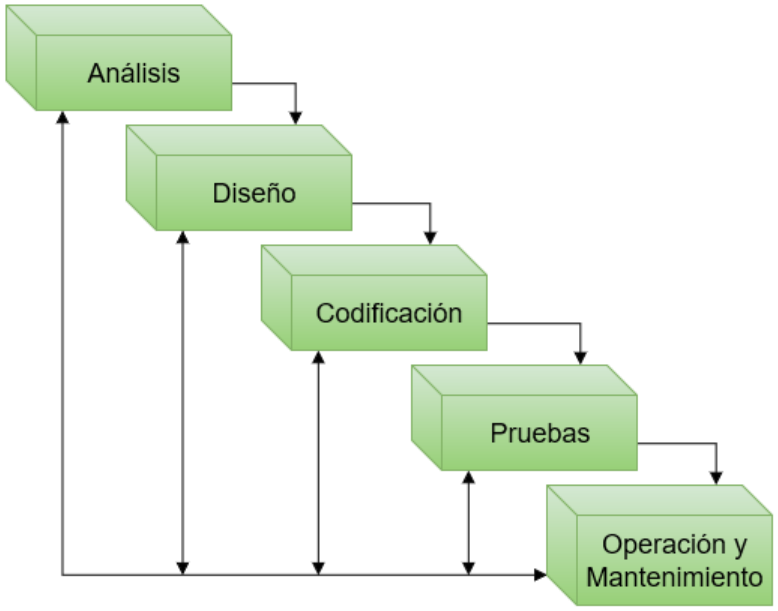


Figura 3-1: Ciclo de vida en cascada con retroalimentación

A continuación, se detallan las tareas realizadas en cada una de las fases del ciclo de vida de este proyecto:

- **Análisis:** Se analizaron las necesidades de los usuarios finales y se realizó un estudio de sistemas de gestión de comunidades disponibles en el mercado con el fin de determinar los objetivos a cubrir. Se realizó un estudio de tecnologías de desarrollo web para la selección de aquella que más se ajustase a las necesidades del proyecto y se especificaron los requisitos funcionales y no funcionales [ANEXO A].

- **Diseño:** En esta fase se diseñó el diagrama Entidad-Relación para la base de datos, el diagrama de casos de uso y un diagrama de secuencia.

- **Codificación:** Implementación de la base de datos y desarrollo de la aplicación web.

- **Pruebas:** Se llevaron a cabo distintos tipos de pruebas para ver el comportamiento del sistema y se realizó una evaluación de la aplicación tras estas actividades.

- **Mantenimiento:** Esta fase no se ha llevado a cabo durante el desarrollo del proyecto. Es la fase destinada a corregir errores e introducir mejoras a lo largo de la explotación del sistema.

3.2 Base de datos

Para el almacenamiento de la información se ha empleado PostgreSQL, para disponer de una base de datos relacional que guarda todos los datos relacionados con comunidades, usuarios, incidencias, anuncios, reuniones, votaciones, gastos y mensajes. En los siguientes apartados se muestra el Diagrama Entidad-Relación y una descripción de las entidades de la base de datos.

3.2.1 Diagrama Entidad-Relación

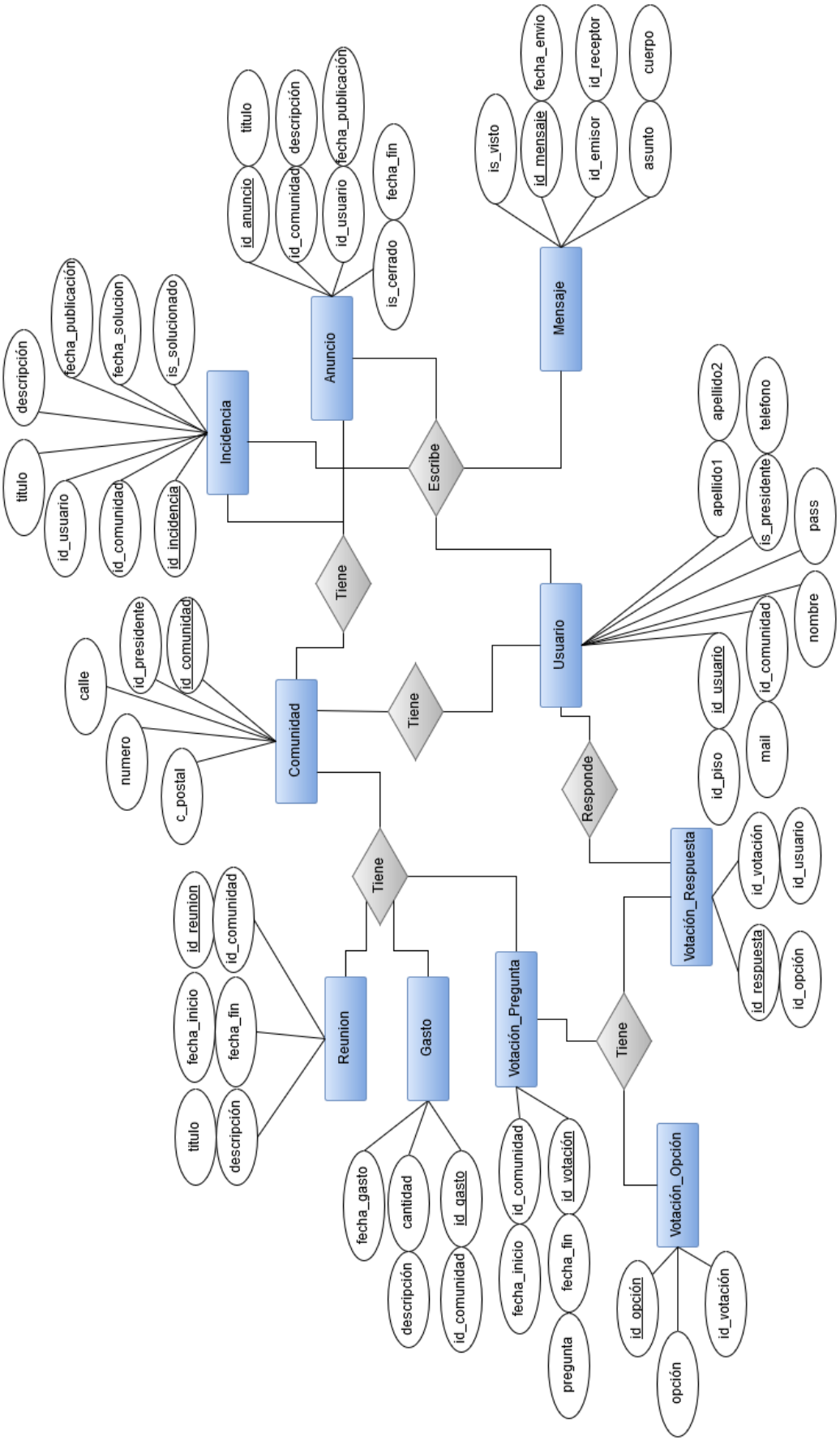


Figura 3-2: Diagrama Entidad-Relación

3.2.2 Entidades de la base de datos

En la Figura 3-2, se muestra el diagrama Entidad-Relación correspondiente a la base de datos diseñada e implementada sobre PostgreSQL. A continuación, se describen las entidades de la base de datos:

- **Comunidades:** Entidad que representa las comunidades de vecinos. Todas deben tener al menos un vecino, que es el presidente.

- **Usuarios:** Entidad que representa a todos los vecinos, se dispone de un atributo de tipo booleano (*is_presidente*) el cual indica si es el usuario es presidente de su comunidad. La contraseña *pass* se encripta con md5. Los usuarios solo pueden pertenecer a una comunidad de vecinos.

- **Incidencias:** Entidad que gestiona los datos relacionados con las incidencias publicadas por los usuarios, las comunidades pueden tener un número ilimitado de incidencias.

- **Anuncios:** Entidad que gestiona los datos relacionados con los anuncios publicados por los usuarios, las comunidades pueden tener un número ilimitado de anuncios.

- **Reuniones:** Entidad que gestiona los datos relacionados con las reuniones publicadas por los presidentes, las comunidades pueden tener un número ilimitado de reuniones.

- **Gastos:** Entidad que gestiona los datos relacionados con los gastos producidos en las comunidades, las comunidades pueden tener un número ilimitado de gastos.

- **Mensajes:** Entidad que gestiona los datos relacionados con los mensajes enviados y recibidos por los usuarios.

- **Votacion_Pregunta:** Entidad que gestiona los datos relacionados con las preguntas de las votaciones, las comunidades pueden tener un número ilimitado de votaciones.

- **Votacion_Opcion:** Entidad que gestiona los datos relacionados con las posibles respuestas a las preguntas de las votaciones, las votaciones pueden tener un máximo de 4 opciones y un mínimo de 2.

- **Votacion_Resultado:** Entidad que gestiona los datos relacionados con las respuestas a las votaciones, una respuesta está relacionada con la opción seleccionada, la votación a la que pertenece y el usuario que realizó la votación.

3.3 Diagrama de casos de uso

El diagrama de casos de uso representa gráficamente la forma en la que los actores operan con el sistema, indicando los pasos que deben realizarse para llevar a cabo alguna actividad.

En este caso como actores tenemos a dos tipos de usuarios, vecino y presidente. El usuario presidente se diferencia del usuario vecino por disponer de más

prestaciones. En la siguiente figura se muestra el diagrama de casos de uso de la aplicación:

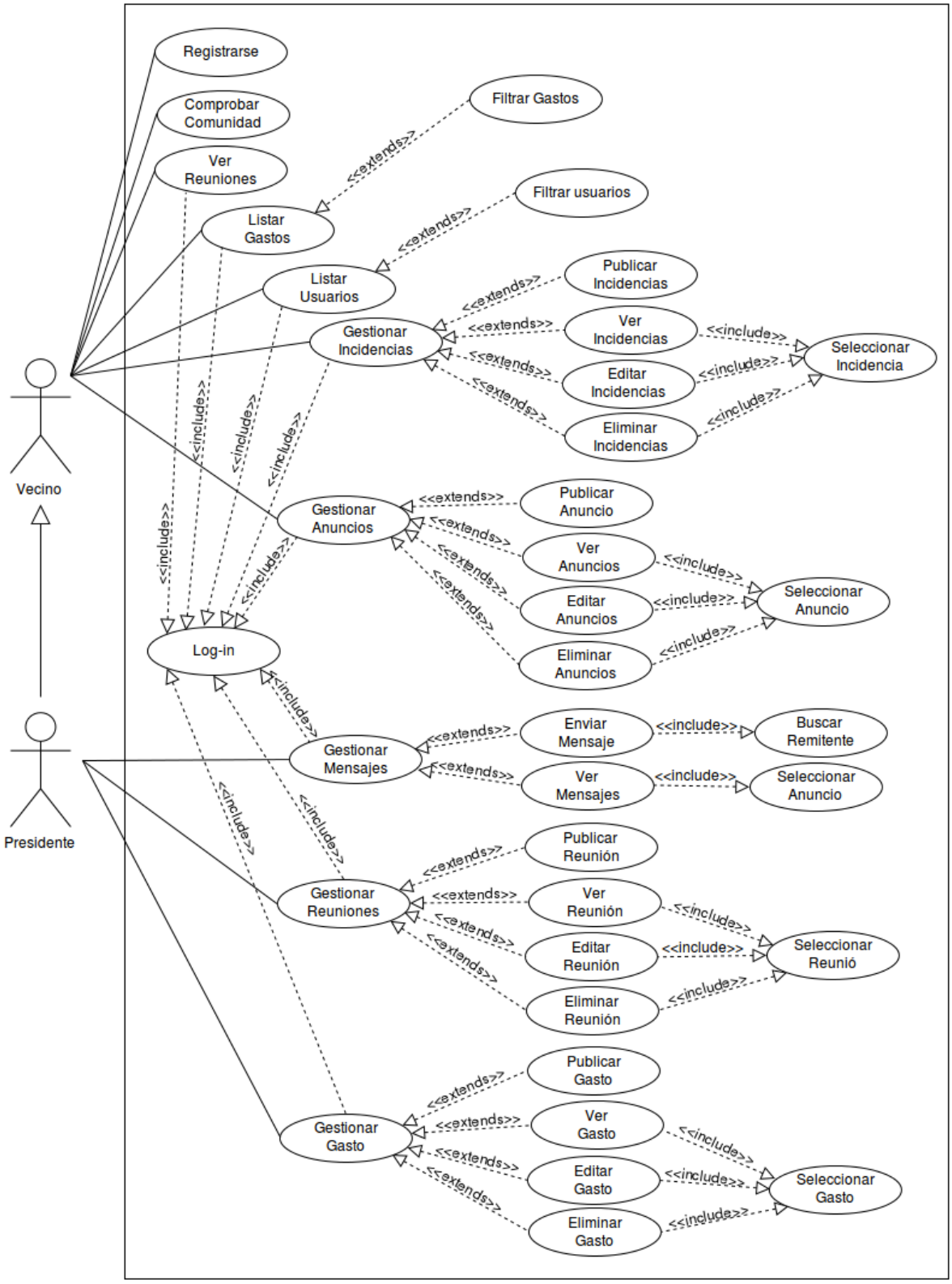


Figura 3-3: Diagrama de Casos de Uso

3.4 Diagrama de secuencia

A continuación, se muestra un diagrama de secuencia de uno de los principales módulos de la aplicación, mostrando las funcionalidades disponibles. La mayoría de los módulos disponen de funcionalidades similares y siguen el mismo diagrama de secuencia, por lo que se ha considerado mostrar uno. En la siguiente figura se muestra el diagrama de secuencia de las funcionalidades disponibles en el módulo de incidencias:

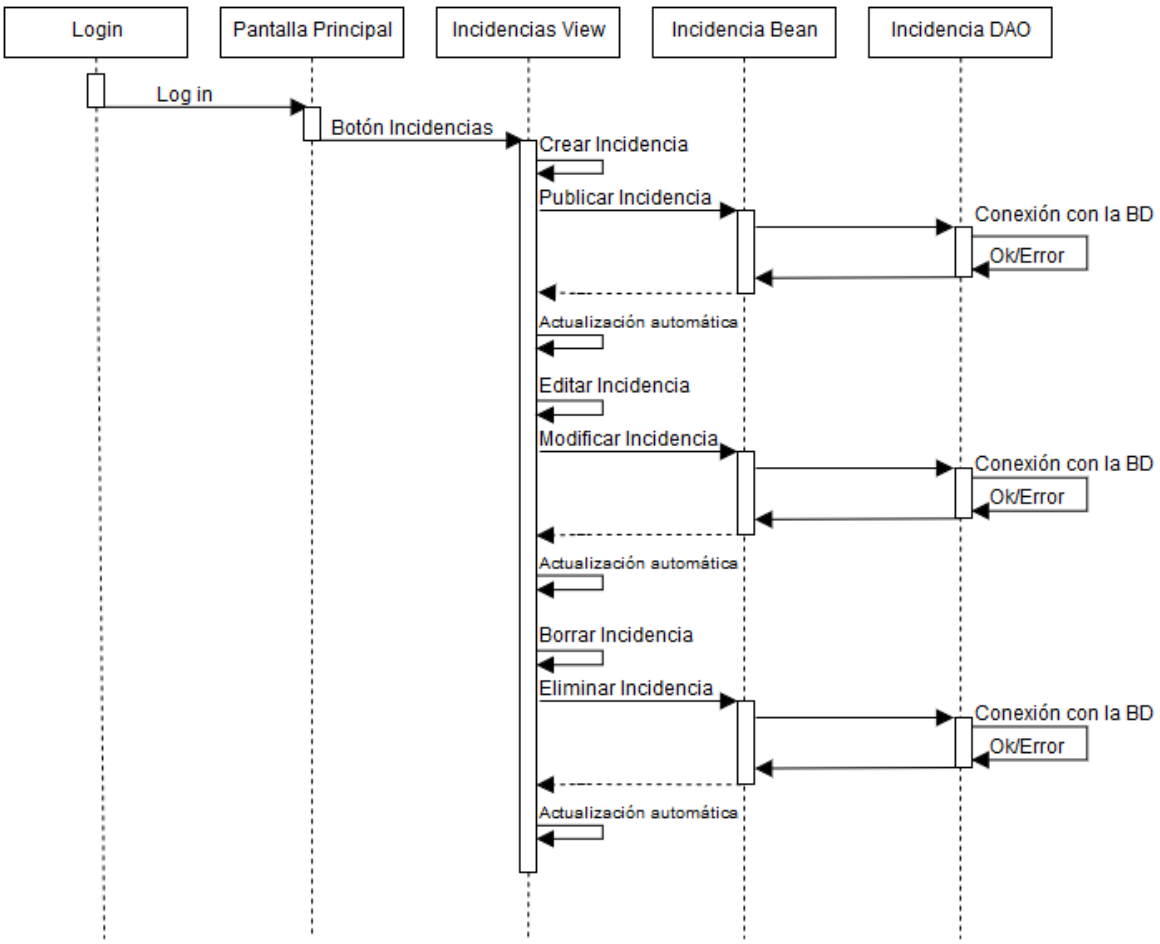


Figura 3-4: Diagrama de Secuencia de Incidencias

Para acceder a los módulos el usuario debe loguearse en la aplicación y una vez dentro seleccionar del menú el módulo al que desea acceder. Dentro del módulo de incidencias, si desea crear una incidencia el usuario pulsa el botón correspondiente con esta acción y se abre un diálogo en el cual se debe introducir la información. Una vez rellenados los campos se pulsa el botón que se encuentra en el interior del diálogo y este evento llama al bean que será el encargado de invocar al método correspondiente del modelo que será quien realizará la inserción en la base de datos. Una vez realiza esta acción el dialogo se cierra y se actualiza de manera automática el panel que muestra las incidencias.

Para modificar una incidencia existente se pulsa el botón de editar y este evento abre un diálogo mostrando el contenido actual del objeto, una vez modificados los datos se pulsa el botón del interior del diálogo y este evento llama al bean que será el encargado de invocar al método correspondiente del modelo que será quien realizará la actualización en la base de datos sobre el objeto modificado. Una vez realiza esta acción el dialogo se cierra y se actualiza de manera automática el panel que muestra las incidencias.

Para eliminar una incidencia se pulsa el botón de eliminar y este evento abre un diálogo de confirmación, si se confirma la acción de eliminar este evento llama al bean que será el encargado de invocar al método correspondiente del modelo que será quien realizará la acción de borrado del objeto de la base de datos. Una vez realiza esta acción el dialogo se cierra y se actualiza de manera automática el panel que muestra las incidencias.

4 Desarrollo

En este capítulo se describen los detalles del proyecto relativos a la fase de desarrollo, que corresponde con la implementación y documentación de la aplicación. Como se puede observar en el estudio realizado en el capítulo Estado del arte, se tomó la decisión de desarrollar la aplicación bajo el framework JSF, implementando de este modo la lógica de la aplicación en Java y haciendo uso del lenguaje SQL para las consultas a la base de datos. Para ello, tras la elección de las tecnologías que serán empleadas se seleccionó como plataforma de desarrollo NetBeans.

4.1 Patrón Modelo-Vista-Controlador (MVC)

La aplicación web ha seguido el patrón de arquitectura software MVC, que se caracteriza por separar las de datos y la lógica de negocio de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones [16].

- **Modelo:** En este caso el modelo es un Sistema de Gestión de Bases de Datos (SGBD), es la capa donde se trabajan los datos. De esta forma, el modelo es el encargado de gestionar todos los accesos a datos, tanto consultas como actualizaciones. En esta aplicación los modelos son los EJBs.

- **Vistas:** Accede a los datos a través del modelo y especifica cómo deben representarse para que los usuarios puedan interactuar con estos. En esta aplicación las vistas son páginas XHTML.

- **Controlador:** Es el responsable de recibir lo eventos de entrada, que son las acciones del usuario, desde la interfaz de usuario, y modificar la vista y el modelo en caso necesario. En esta aplicación los controladores son los Managed Beans.

Esta disposición de los componentes del proyecto potencia la facilidad del mantenimiento, reutilización de código y la separación de conceptos, haciendo el software más robusto y permitiendo el desarrollo en paralelo de los distintos componentes.

A continuación, se muestra una representación gráfica de este patrón de diseño en la que se muestran las distintas capas y la relación entre ellas:

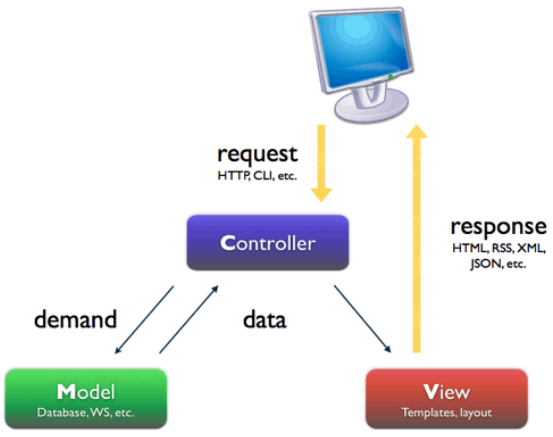


Figura 4-1: Patrón MVC

El código se ha desarrollado por módulos, de manera que cada uno de estos dispone de su correspondiente modelo, vista y controlador. Los principales módulos de la aplicación son los siguientes: incidencias, anuncios, usuarios, reuniones, votaciones, gastos y mensajes.

4.2 Modelo: Enterprise JavaBeans

El desarrollo de los modelos en el proyecto se ha realizado por medio de Beans empresariales J2EE (Enterprise JavaBeans - EJB) y JDBC. EJBs son componentes del lado del servidor que encapsulan la lógica de negocio y se encargan de las transacciones y la seguridad. La lógica de negocio es el conjunto de reglas que se siguen en el software para reaccionar a distintos eventos y situaciones. En este proyecto se han empleado EJB de sesión sin estado (Stateless), se caracterizan por no almacenar los datos de los clientes que los invocan y cada llamada es independiente de las demás.

En este proyecto los contenedores EJBs son los encargados de realizar las operaciones con la base de datos y se usa conjuntamente con JDBC dado que esta es la tecnología encargada de realizar la conexión con la base de datos relacional.

Para declarar una clase como EJB basta con añadir la anotación correspondiente, *@EJB*. Cada entidad de la base de datos posee su correspondiente EJB bajo el patrón de diseño DAO. Este patrón separa la lógica de negocio y la responsabilidad de la persistencia de los datos. La ventaja de esto es que cualquier cambio en la capa de persistencia no afecta a la lógica de negocio. En la siguiente figura se puede observar representado este patrón de diseño:

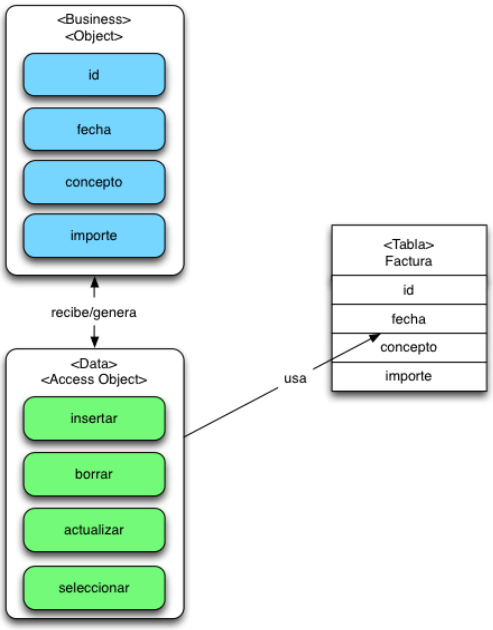


Figura 4-2: Ejemplo de Patrón DAO

En las clases DAO se encuentran todas las querys necesarias para llevar a cabo las prestaciones que ofrece la aplicación. Las querys son declaradas como constantes de la clase y después se encuentran los métodos encargados de ejecutarlas y devolver lo que obtienen tras su ejecución.

4.2.1 Ejemplo de código: Modelo de Usuarios

A continuación, se muestra un ejemplo de declaración de la clase como EJB y la declaración de las querys como constantes:

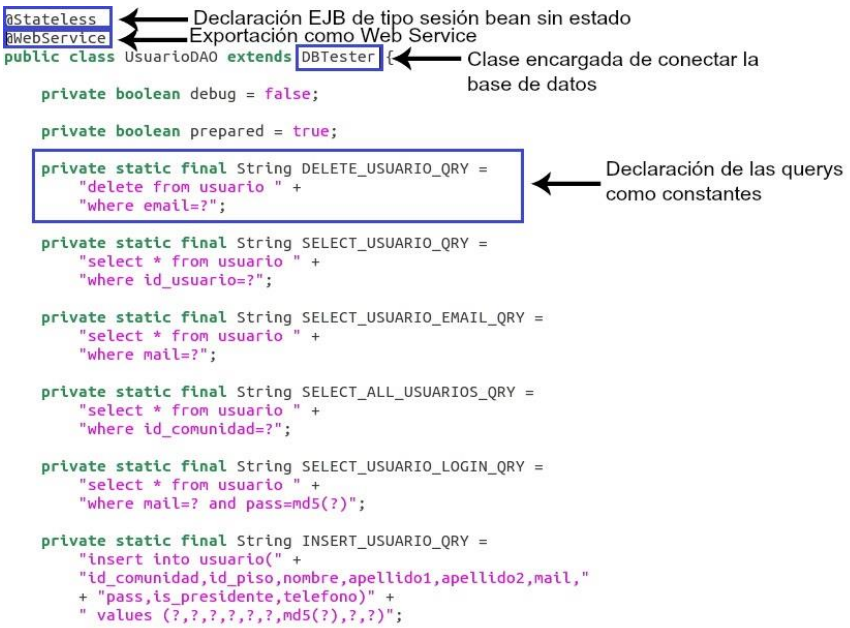


Figura 4-3: Modelo de Usuarios - Declaraciones

En la siguiente figura se muestra el método que se invoca cuando se desea buscar un usuario por su identificador:

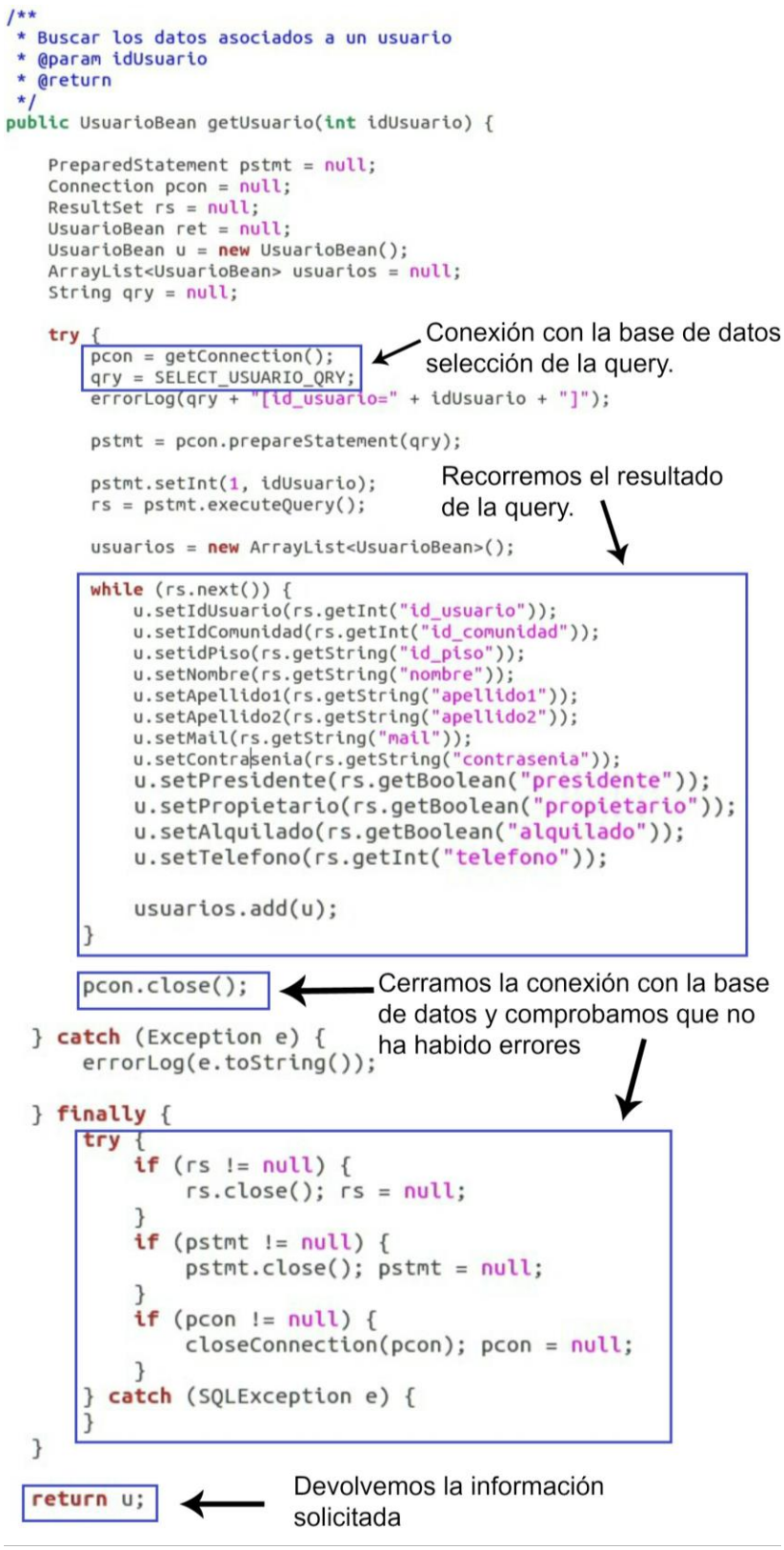


Figura 4-4: Modelo de Usuarios – Método

4.3 Vista: Páginas XHTML

El desarrollo de la vista en JSF se ha realizado por medio de facelets y XHTML. Facelets es un lenguaje de declaración de páginas usado para construir vistas de JSF usando plantillas de estilo HTML y para construir árboles de componentes. Las principales

ventajas de usar esta tecnología para la implementación de las vistas en lugar de otras es que Facelets reduce el tiempo y el esfuerzo que se necesita para el desarrollo y despliegue de aplicaciones [17].

Las vistas facelets suelen ser creadas como páginas XHTML. XHTML es código HTML escrito como XML y está pensado para sustituir a HTML como estándar de desarrollo de páginas web. Es más estricto a nivel técnico dado que hereda la rigidez de XML y es más robusto [18].

Durante el desarrollo de la interfaz se ha tenido muy en cuenta que la aplicación va dirigida a un público muy diverso, entre el cual puede haber usuarios con bajos conocimientos de navegación web. Por esta razón, se ha optado por una interfaz y un flujo de navegación intuitivo y sencillo, mostrando varios módulos la misma forma de interacción con lo que se pretende que la fase de aprendizaje y manejo de la aplicación por parte de los usuarios sea la más breve posible y puedan aprovechar todas las funcionalidades que esta les ofrece.

Las vistas acceden a los controladores de la siguiente forma *#{bean.atributo}* o *#{bean.metodo()}*.

4.3.1 Plantillas de estilo

El uso de plantillas en JSF es muy sencillo y cómodo para el desarrollador, ya que evita tener código duplicado en múltiples páginas teniendo en cuenta que la mayoría de las vistas tendrán un estilo común en la cabecera, el menú o el pie de página. Por lo tanto, estos elementos se definen en la plantilla y se dejan vacíos y para cada nueva página solo habrá que indicar la plantilla que se desea utilizar y el contenido de cada elemento vacío de la plantilla.

En este proyecto se han creado cuatro plantillas: para la pantalla de inicio, para la página de login, para las demás páginas que el usuario puede visualizar sin estar logueado en la aplicación y otra para cuando ya se encuentra dentro de ésta.

4.3.2 Librerías

Como ya se ha comentado antes, las vistas en JSF utilizan etiquetas de alto nivel para declarar los componentes de éstas, que pueden ser tan sencillos como un botón o una entrada de texto o complejos como tablas y gráficos. JSF dispone de multitud de bibliotecas y en este proyecto se ha hecho uso de cuatro de ellas:

- **Biblioteca Facelets (xmlns:ui="http://xmlns.jcp.org/jsf/facelets"):** Contiene etiquetas específicas de facelets, entre las prestaciones de estas etiquetas destacamos su función de crear las plantillas y gestionar las secciones de estas. Las etiquetas más usadas de esta biblioteca han sido: *composite*, para declarar la plantilla que usará la página; *define*, para declarar el contenido de una sección de la plantilla; y *repeat*, para la declaración de un bucle, con lo que podemos iterar entre los elementos de un array o lista.

- **Biblioteca HTML (xmlns:h="http://xmlns.jcp.org/jsf/html"):** Esta biblioteca sirve para renderizar componentes propios de HTML. Las etiquetas más usadas de esta biblioteca han sido: *form*, *outputText* para el texto de salida, *inputText* para el

texto de entrada, *panelGrid* corresponde a las tablas de HTML, *panelGroup* y *selectOneMenu* para la selección de una opción entre varias dadas.

- **Biblioteca Primefaces (xmlns:p="http://primefaces.org/ui"):** Estas biblioteca contiene una gran cantidad componentes de alto nivel. Las etiquetas más usadas de esta biblioteca han sido: *dialog*, como diálogo emergente para editar y crear contenido, para confirmar una acción o para ver información detallada sobre un elemento; *Ajax*, para la gestión de eventos y principalmente para la actualización automática de componentes tras la modificación de estos; *inputTextarea*, para texto de entrada de grande longitud; *panel*, para mostrar datos en paneles desplegables; *growl*, para mostrar mensajes informativos emergente en el lateral de la pantalla; *chart*, para mostrar los gráficos de gastos y para mostrar el porcentaje en las votaciones; *dataTable*, para mostrar tablas con interfaces complejas; *tabView* y *tab*, para separar los datos en distintas pestañas; *commandButton*, para la declaración de botones; y *Schedule*, etiqueta proporciona el calendario donde se indican las reuniones de la comunidad. Algunos de estos componentes han sido modificados por medio de código JavaScript para personalizarlos, como el calendario.

- **Bilbioteca core (xmlns:f="http://xmlns.jcp.org/jsf/core"):** Contiene etiquetas de carácter transversal y entre las funcionalidades de éstas destacamos la manipulación de eventos, conversión de datos y validación de datos. Se han usado pocas etiquetas de esta biblioteca dado que hay otras bibliotecas con las mismas etiquetas que ofrecen mejores prestaciones. Esto sucede por ejemplo con la etiqueta ajax que se encuentra también en la biblioteca Primefaces. Las etiquetas más usadas de esta biblioteca han sido: *facet*, para poner títulos; *validator*, para la validación de campos vacíos; y *selectItems*, para la selección de una opción entre varias dadas.

4.3.3 Ejemplo de código: Vista de Usuarios

Figura 4-5: Vista de Usuarios – Parte 1

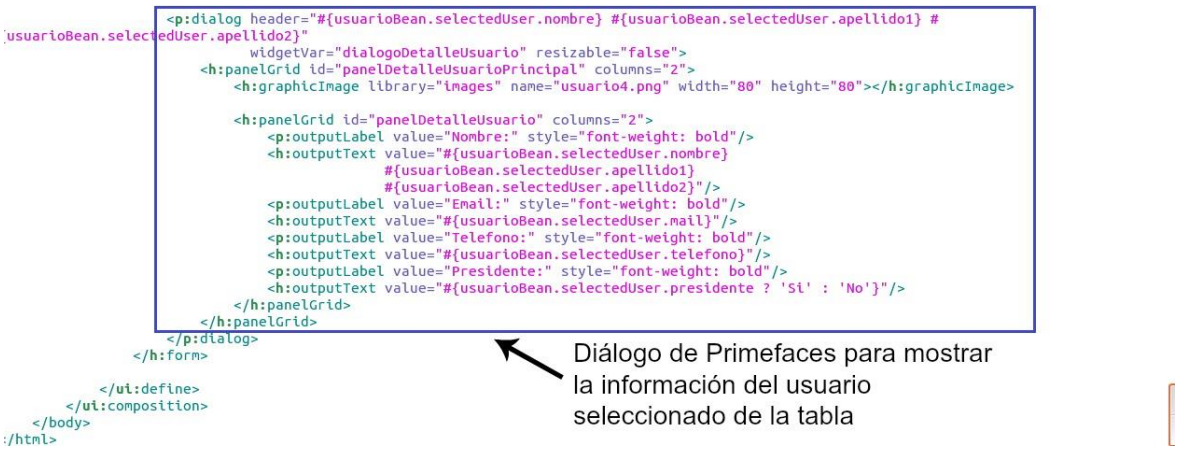


Figura 4-6: Vista de Usuarios – Parte 2

4.4 Controlador: Managed Beans

El desarrollo de los controladores en proyecto se ha realizado con Managed Beans, que son los encargados de recibir los eventos, invocar al modelo y cambiar la vista.

Los Managed Beans son clases Java Bean creadas y gestionadas bajo la tecnología JSF. Por lo tanto, son clases Java con un constructor sin argumentos. Los atributos de la clase son privados y disponen de sus correspondientes métodos get/set. La declaración de éstos se realiza mediante la anotación *@ManagedBean* y además también hay que añadirle una anotación que indique su alcance. Estas anotaciones siempre van después de la anotación que indica que es un Managed Bean. Hay cuatro tipos de alcance:

- **@RequestScoped:** El bean persiste solo durante la petición del usuario, y es el alcance por defecto si no se especifica ningún otro.

- **@ViewScoped:** El bean persiste mientras que el usuario se encuentre en la misma vista, y es el recomendado si se hace uso de ajax.

- **@SessionScoped:** El bean persiste durante el tiempo de la sesión del usuario.

- **@ApplicationScoped:** El bean persiste durante toda la aplicación.

Cuando el usuario interacciona con la aplicación los controladores reciben estos eventos, y si alguno de estos eventos requiere la modificación o la obtención de información almacenada en la base de datos, hay que llamar al correspondiente método del EJB, pasarle la información necesaria para llevar a cabo la tarea y esperar a recibir la respuesta del modelo. Las anotaciones *@EJB* en estas clases indican una dependencia a un EJB.

Algunos Managed Beans disponen de un método precedido por la anotación @PostConstruct, esto indica que ese método debe ejecutarse tras la construcción del bean. Esto solo sucede una vez en el ciclo de vida de JSF, al igual que la creación del vean. No cada vez que se abra la página.

4.4.1 Ejemplo de código: Controlador de Usuarios

A continuación, se muestran las declaraciones realizadas en el controlador de los usuarios:

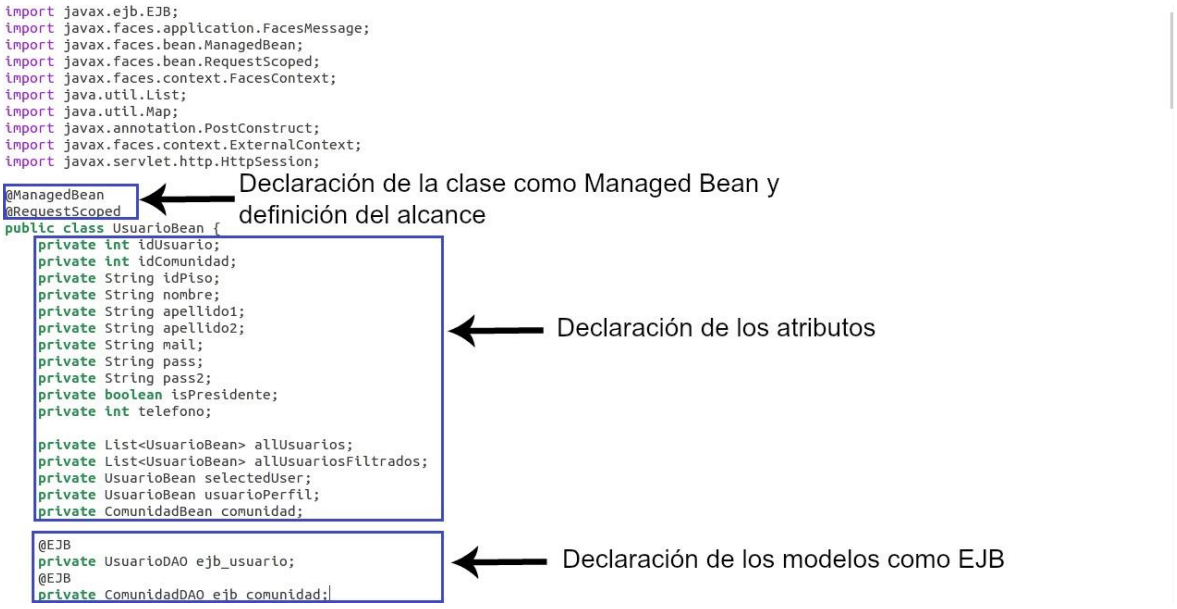


Figura 4-7: Controlador de Usuarios – Declaraciones

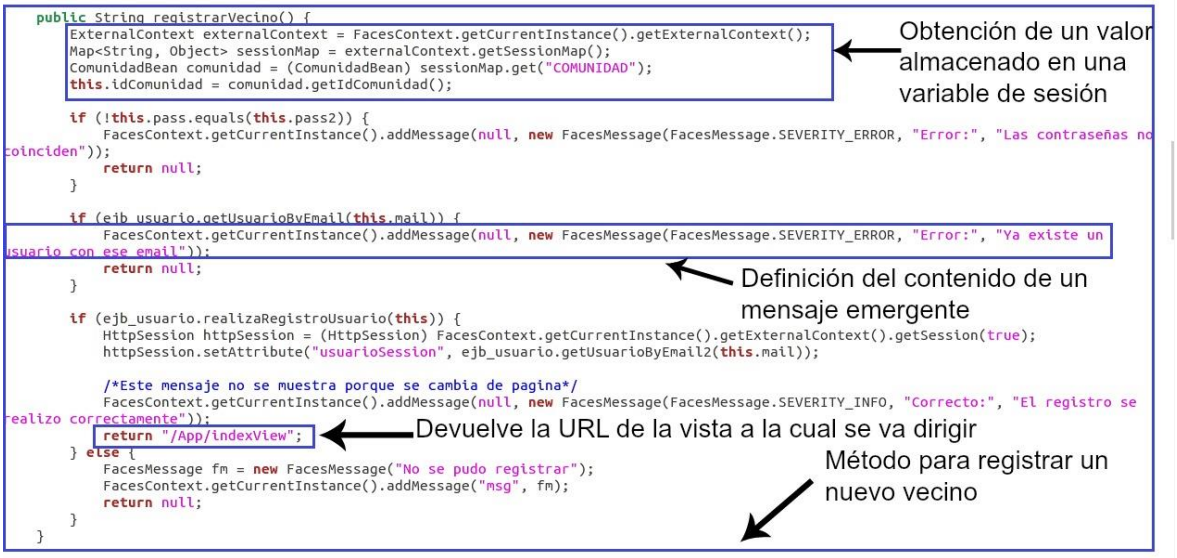


Figura 4-8: Controlador de Usuarios – Método

4.5 Filtro de Sesión

Se ha implementado un filtro de sesión para controlar y bloquear el acceso cuando un usuario que no haya realizado el login intente acceder a aquellas páginas de la aplicación en las sea necesario estar logueado. El filtro redirecciona al usuario a la página de Login cuando este intenta acceder a una URL no permitida o cuando la sesión haya sido invalidada, esto último sucede cuando hay un periodo de inactividad superior a 30 minutos.

Para definir el filtro es necesario añadir la anotación *@WebFilter* en la clase que lo implementa, y contiene metadatos sobre el filtro que se declaró. En la anotación se debe

especificar al menos un patrón URL, en este caso se ha especificado el patrón URL de las vistas a las cuales solo se debe acceder una vez logueado [19].

Esta clase implementa la interfaz javaz.servlet.Filter, que contiene 3 métodos:

- **Init:** Este método es llamado cuando el filtro es inicializado en el contenedor, esto sucede cuando se realiza un deploy.

- **Destroy:** Este método es llamado cuando el filtro es destruido en el contenedor.

- **doFilter:** Este método es el más importante, dado que es el encargado de realizar toda la actividad del filtro. Aquí se indica la dirección a la que se debe redireccionar a los usuarios.

A continuación, se muestra una imagen del filtro implementado:


```
/**
 *
 * @author Silvia Nerea Anguita de Blas
 *
 */
@WebFilter("/faces/App/*")
public class SessionUrlFilter implements Filter{

    FilterConfig filterConfig;

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
        this.filterConfig = filterConfig;
    }

    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
        HttpServletRequest req = (HttpServletRequest) request;
        HttpServletResponse res = (HttpServletResponse) response;
        HttpSession session = req.getSession(true);

        String requestUrl = req.getRequestURL().toString();

        if(session.getAttribute("usuarioSession") == null && !requestUrl.contains("LoginView")){
            String url = req.getContextPath()+"/Login/LoginView";
            res.sendRedirect(req.getContextPath());
        } else{
            chain.doFilter(request, response);
        }
    }

    @Override
    public void destroy() {
        this.filterConfig = null;
    }

}
```

Figura 4-9: Filtro de Sesión

El periodo de inactividad durante el cual la sesión sigue estando activa se declara en el fichero *web.xml*, en la siguiente imagen podemos observar cómo se declara:

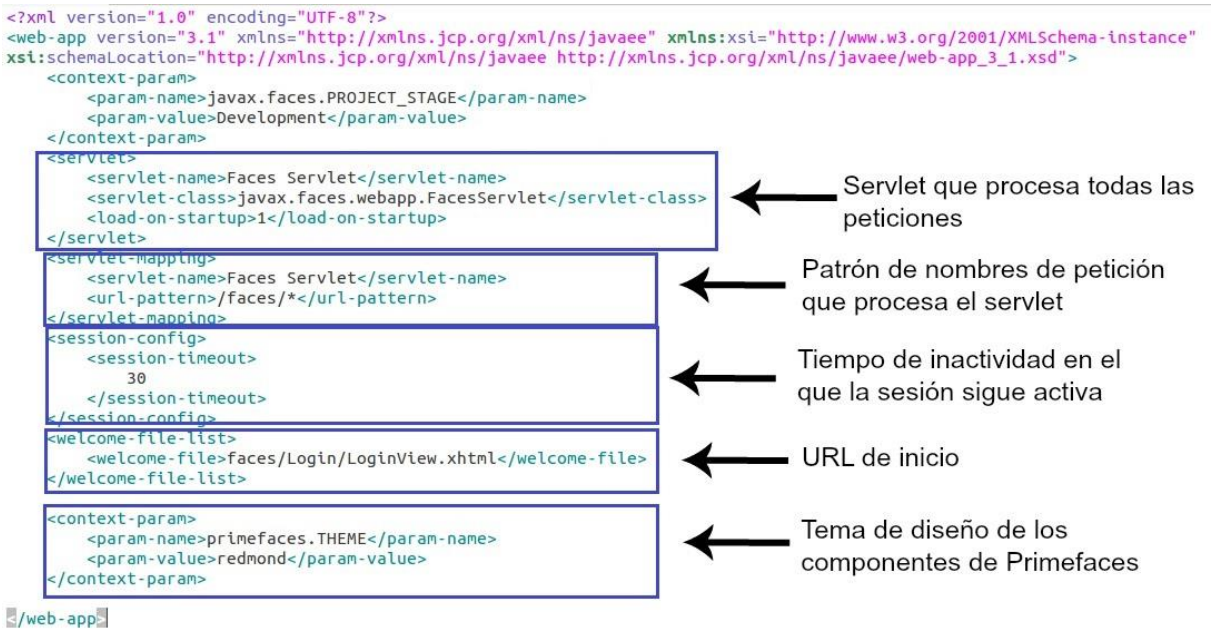
```
<session-config>
    <session-timeout>
        30
    </session-timeout>
</session-config>
```

Figura 4-10: Periodo de inactividad

4.6 Fichero web.xml

El fichero *web.xml* es un descriptor de despliegue, su función es describir cómo se debe desplegar la aplicación web. En este fichero se ha declarado el servlet que procesa todas las peticiones, el patrón de nombres de petición que procesa el servlet, la página de bienvenida

de la aplicación, el tiempo de inactividad válido en una sesión y el tema de diseño de los componentes Primefaces. En la siguiente imagen se pueden observar todas las declaraciones:




```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
    <context-param>
        <param-name>javax.faces.PROJECT_STAGE</param-name>
        <param-value>Development</param-value>
    </context-param>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>/faces/*</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>faces/Login/LoginView.xhtml</welcome-file>
    </welcome-file-list>

    <context-param>
        <param-name>primefaces.THEME</param-name>
        <param-value>redmond</param-value>
    </context-param>
</web-app>
```

Figura 4-11: Fichero web.xml

5 Integración, pruebas y resultados

Debido al ciclo de vida empleado, la integración y las pruebas no fueran realizas de manera global al finalizar la codificación. Sino que fue un proceso que se llevó a cabo junto con la codificación. Al finalizar la codificación de un módulo se realizaban las pruebas correspondientes para comprobar su correcto funcionamiento. Una vez finalizado el desarrollo de la aplicación se comprobó el correcto funcionamiento de todos los módulos.

Debido a la gran cantidad de módulos y a la similitud en su forma de actuar, se van a representar las pruebas realizadas para el módulo de incidencias, el módulo de gastos y el acceso a la aplicación ya que es un paso indispensable para poder acceder a estas funcionalidades.

5.1 Prueba de Login

Lo primero que muestra la aplicación es una pantalla de inicio, donde se describen los principales servicios prestados. Se debe pulsar "Accede al contenido" para dirigirnos al Login.

Figura 5-1: Pantalla de Inicio

La pantalla de Login muestra dos campos de texto que nos permiten introducir el mail y la contraseña, teniendo en cuenta que ya estamos registrados y formamos parte del sistema.

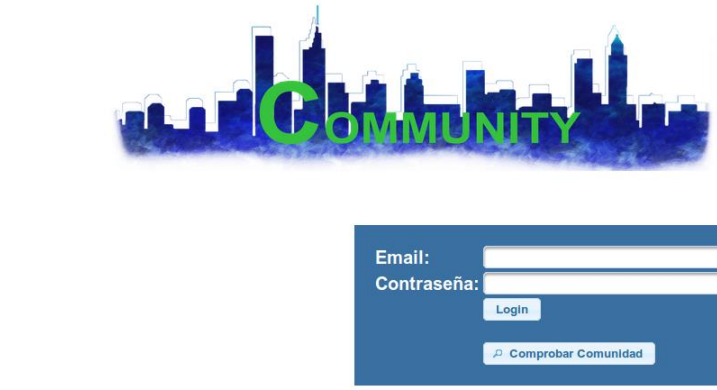


Figura 5-2: Pantalla de Login

Si dejamos algún dato en blanco o los datos introducidos no son los correctos, porque nos hayamos equivocado en el correo o la contraseña. El sistema muestra un mensaje emergente en la parte superior derecha de la pantalla informándonos del error.

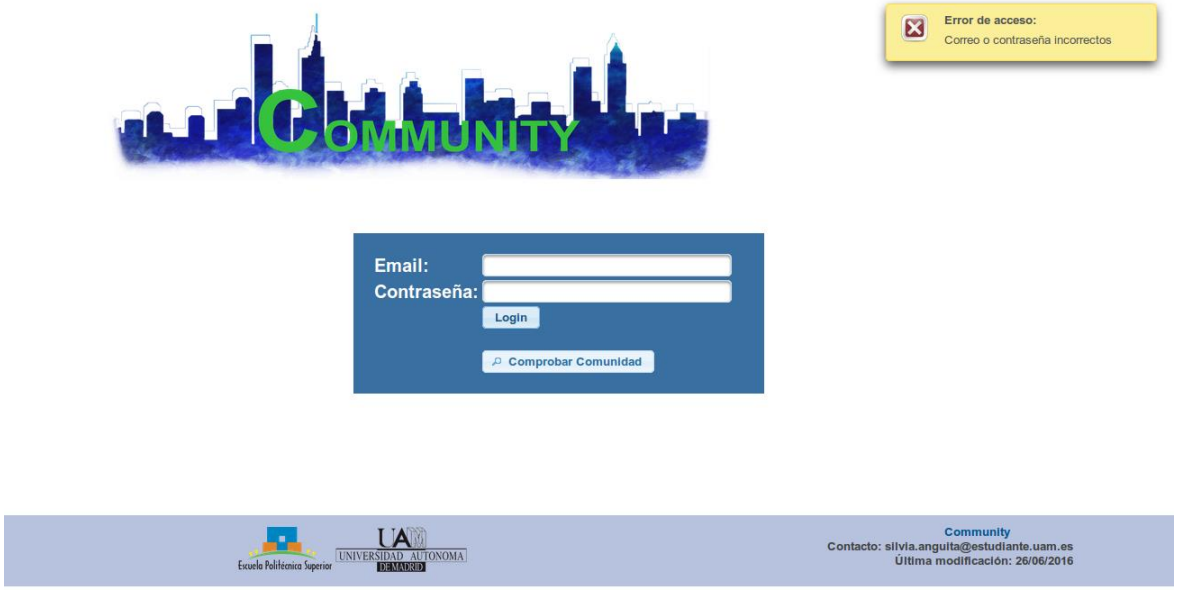


Figura 5-3: Pantalla de Login con los datos en blanco

Suponiendo que accedemos al sistema de forma satisfactoria tras la introducción correcta del correo y la contraseña, se mostrará la sección de inicio del interior de la aplicación. En la parte superior derecha de la pantalla aparece el nombre del usuario logueado, un botón que nos permitirá cerrar la sesión y salir de la aplicación y un mensaje "Tiene mensajes sin leer", que aparece si hay mensajes sin leer en la bandeja de entrada del usuario logueado. En esta sección se muestra un tutorial indicando las funciones de cada uno de los módulos.

Figura 5-4: Sección de Inicio dentro de la aplicación

5.2 Pruebas en el módulo de incidencias

Nada más entrar en la sección de incidencias esta es la vista que tenemos:

Figura 5-5: Pantalla inicial de Incidencias

Si pasamos el ratón por encima del título de la sección aparece un cuadro de texto mostrando ayuda acerca del módulo en el que nos encontramos. Esto se encuentra en todas las secciones.

Esta sección muestra tres paneles en los que se clasifican las incidencias de la comunidad. Las incidencias en color naranja se encuentran en proceso de gestión y las azules son las que ya se han solucionado.

Para añadir una incidencia se debe pulsar el botón que se encuentra en la parte superior derecha y completar todos los campos antes de enviarla. En la siguiente figura se muestra

el caso en el que se intenta enviar una incidencia con algún campo vacío. El diálogo permanecerá abierto y se mostrará un mensaje emergente de indicando el error:

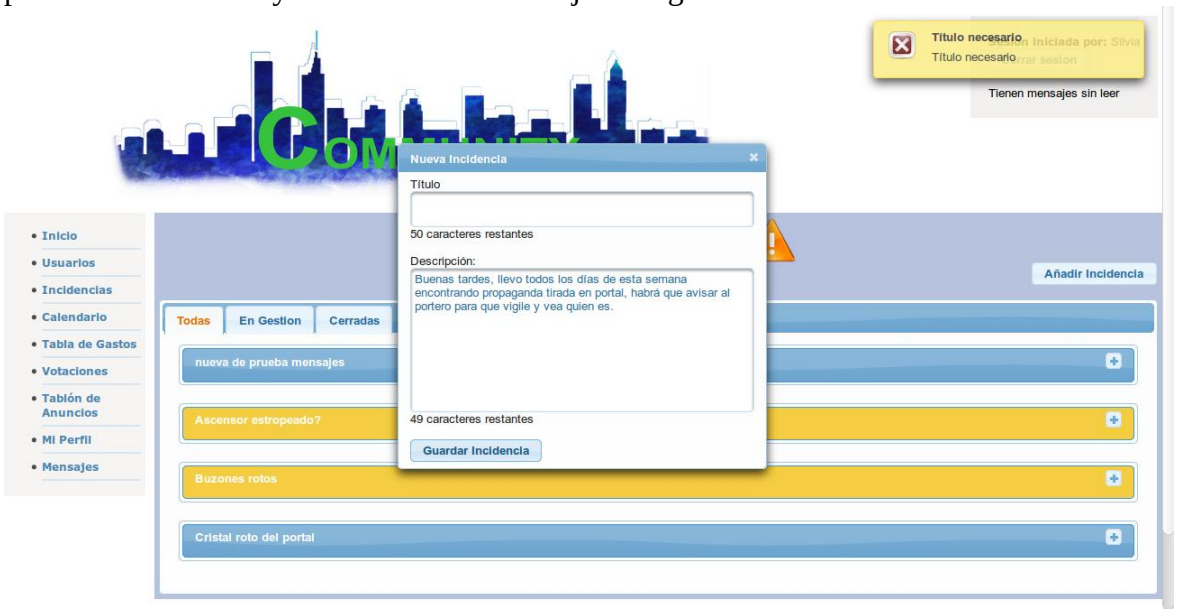


Figura 5-6: Añadir Incidencia con campos vacíos

Lo mismo sucede si intenta editar alguna incidencia existente dejando algún campo vacío:

Figura 5-7: Editar Incidencia con campos vacíos

La aplicación tampoco permite la inserción de una incidencia con el mismo título y descripción que una ya existente:

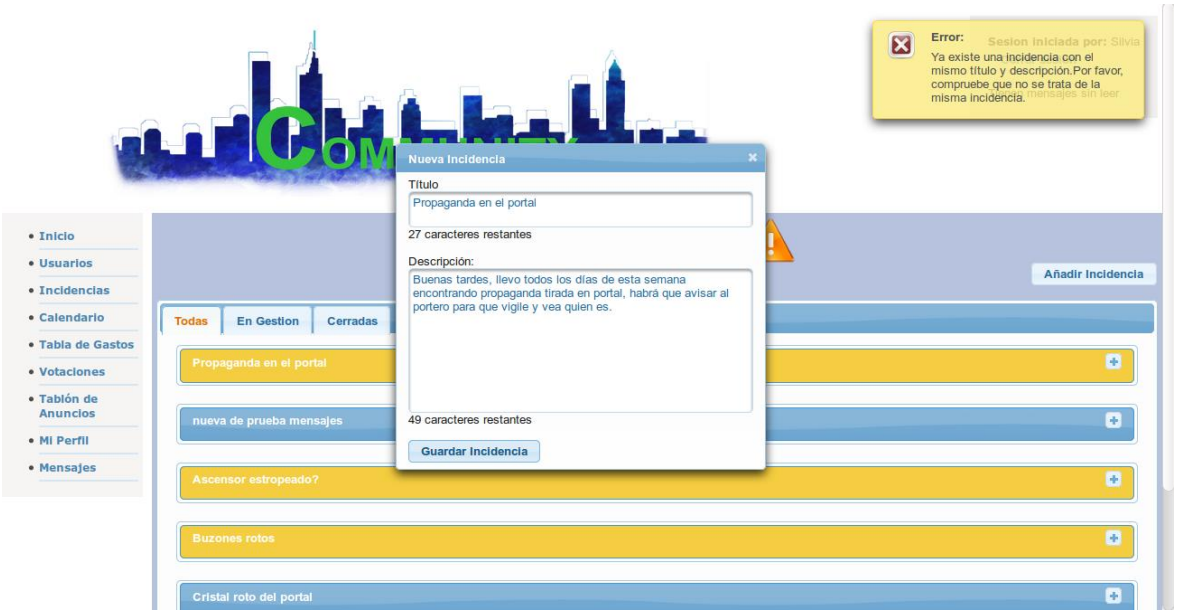


Figura 5-8: Inserción de una Incidencia repetida

Si por el contrario se introducen todos los datos correctamente el diálogo se cerrará y se mostrará un mensaje emergente informándonos que el registro de la incidencia se realizó correctamente:

Figura 5-9: Inserción correcta de la Incidencia

Si queremos borrar una incidencia antes de eliminarse por completo se mostrará un diálogo de confirmación.

Figura 5-10: Diálogo de confirmación de la eliminación

Si procedemos con la acción y la confirmamos la incidencia se borrará desapareciendo automáticamente de la lista y se mostrará un mensaje emergente informándonos que la eliminación de la incidencia se realizó correctamente:

Figura 5-11: Mensaje de eliminación de la incidencia

5.3 Pruebas en el módulo de gastos

Nada más entrar en la sección Tabla de Gastos se muestran los gatos de la comunidad en forma de gráfico de barras y en forma de tabla:

Figura 5-12: Pantalla inicial de Gastos

Para añadir un gasto se debe pulsar el botón que se encuentra en la parte superior derecha y completar todos los campos antes de enviarlo. En la siguiente figura se muestra el caso en el que se intenta enviar un gasto con los campos vacíos. El diálogo permanecerá abierto y se mostrarán mensajes emergentes de indicando los errores:

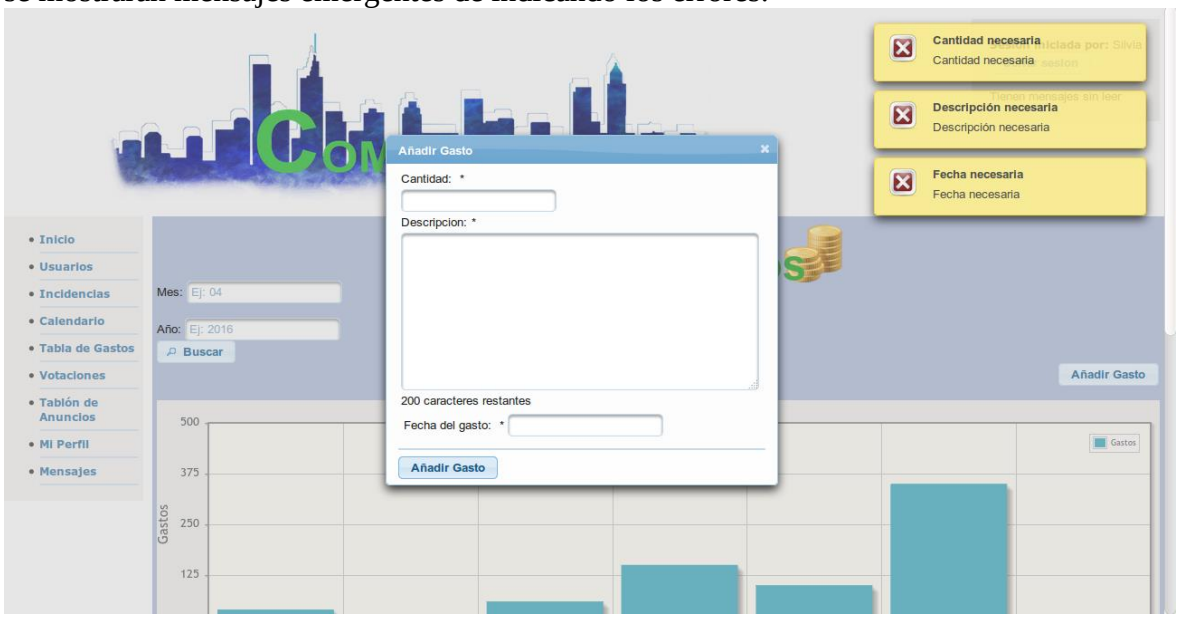


Figura 5-13: Añadir Gasto con campos vacíos

Si por el contrario se introducen todos los datos correctamente el diálogo se cerrará y se mostrará un mensaje emergente informándonos que el registro del gasto se realizó correctamente:

Figura 5-14: Inserción correcta del gasto

Para editar o eliminar un gasto se debe seleccionar un objeto de la tabla. Se abrirá un diálogo mostrando la información del gasto junto con los botones de editar y eliminar. El funcionamiento de estos botones es idéntico a los botones de editar y eliminar de la sección de incidencias:

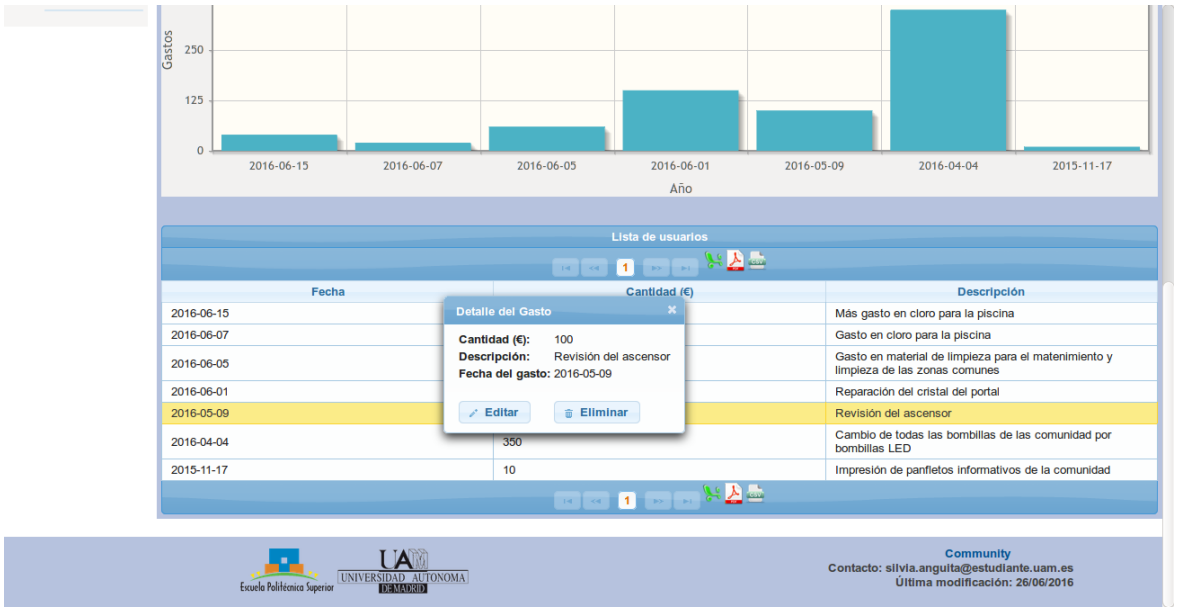


Figura 5-15: Diálogo mostrando detalles de un gasto

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Los resultados obtenidos tras el desarrollo de este TFG han sido satisfactorios, puesto que se ha conseguido que Community sea una aplicación web que cumple con su objetivo principal: permite la comunicación y autogestión de comunidades de vecinos de manera sencilla con una interfaz intuitiva y sencilla.

La aplicación desarrollada cumple perfectamente todas las funcionalidades descritas en los objetivos del proyecto, permitiendo incluso la descarga de la información sobre los gastos en formato Excel, PDF y CSV, y realizando el envío de mensajes automáticos cuando se realiza alguna nueva publicación.

Centrándonos en el aprendizaje, el framework JSF es muy sencillo, debido a la similitud de las páginas XHTML con lenguaje HTML, y facilita el desarrollo de aplicaciones web, que junto con los componentes de Primefaces ofrecen resultados muy buenos.

6.2 Trabajo futuro

Tras el desarrollo del proyecto se ha obtenido una primera versión totalmente operativa y que cumple con los principales objetivos.

A continuación, se listan funcionalidades que se añadirán para la obtención de una aplicación más completa:

- Desarrollo de un foro donde los usuarios podrán debatir distintos temas de interés.
- Incorporación de un servicio de comentarios para las incidencias y los anuncios.
- Incluir un apartado de documentación donde los demás vecinos puedan subir documentos y que estos puedan ser visualizados y descargados por otros usuarios.
- Desarrollo de una versión para dispositivos móviles.
- Ampliar el alcance la aplicación añadiendo más idiomas.
- Incluir la posibilidad de añadir foto de perfil en los usuarios.

Referencias

[1] Sandra López Letón, “Vecinos sin secretos tras el portal”, El País. http://economia.elpais.com/economia/2015/01/08/vivienda/1420726722_170660.html
[2] VecinosenRed.es, web. http://www.vecinosenred.es/
[3] Desarrollo web, Wikipedia. https://es.wikipedia.org/wiki/Desarrollo_web
[4] ASP.NET, Wikipedia. https://es.wikipedia.org/wiki/ASP.NET
[5] PHP, Wikipedia. https://es.wikipedia.org/wiki/PHP
[6] JSP, Wikipedia. http://www.oracle.com/technetwork/java/javaee/jsp/index.html
[7] Ruby on Rails, web. http://www.rubyonrails.org.es/
[8] Ruby on Rails, Wikipedia. https://es.wikipedia.org/wiki/Ruby_on_Rails
[9] Django, web. https://www.djangoproject.com/
[10] Tutorial JSF. http://www.coreservlets.com/JSF-Tutorial/jsf2/
[11] Primefaces, web. http://primefaces.org/
[12] JDBC, Wikipedia. https://es.wikipedia.org/wiki/Java_Database_Connectivity
[13] PostgreSQL, web. https://www.postgresql.org/
[14] CSS3, W3Schools. http://www.w3schools.com/css/css3_intro.asp
[15] AJAX, Wikipedia. https://es.wikipedia.org/wiki/AJAX
[16] MVC, Wikipedia. https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador
[17] The Java EE 6 Tutorial, Facelets. http://docs.oracle.com/javaee/6/tutorial/doc/gijtu.html
[18] XHTML, Wikipedia. https://es.wikipedia.org/wiki/XHTML
[19] The Java EE 6 Tutorial, Filters. http://docs.oracle.com/javaee/6/tutorial/doc/bnagb.html

Glosario

API	Application Programming Interface.
JSP	JavaServer Pages, tecnología orientada a crear páginas web con programación en Java.
JSF	JavaServer Faces, tecnología y framework para aplicaciones Java basadas en web.
PHP	Hypertext Preprocessor, lenguaje de programación del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.
RoR	Ruby on Rails, framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby.
EJB	Enterprise JavaBeans, una de las API que forman parte del estándar de construcción de aplicaciones empresariales J2EE.
JDBC	Java DatabaseConectivity, API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java.
CSS3	Cascading Style Sheets 3, lenguaje usado para definir y crear la presentación de un documento estructurado.
AJAX	Asynchronous JavaScript And XML, técnica de desarrollo web para crear aplicaciones interactivas.
SQL	Structured Query Language, lenguaje declarativo de acceso a bases de datos.
XHTML	Application Programming Interface, es una versión más estricta y limpia de HTML.
MVC	Model-View-Controller, patrón de arquitectura de software.

Anexos

A Requisitos Funcionales y No Funcionales

La aplicación cuenta con dos usuarios, uno general que serán los vecinos y otro con ciertos privilegios que será asignado al presidente de la comunidad.

Requisitos Funcionales de los usuarios:

- **RF-1.** La aplicación contará con un apartado en el cual los usuarios podrán comprobar si su comunidad ya se encuentra registrada

- **RF-2.** Los usuarios podrán darse de alta una comunidad cuando esta no esté registrada y darse de alta como vecinos, o como presidente en caso de que la comunidad no se encontrase registrada.

- **RF-3.** La aplicación permitirá a los usuarios loguearse, mediante la introducción de su email y contraseña.

- **RF-4.** La aplicación permitirá a los usuarios cerrar su sesión.

- **RF-5.** Los usuarios podrán ver la lista de vecinos de la comunidad y ver información de contacto sobre estos.

- **RF-6.** La aplicación dispondrá de una sección de incidencias donde los usuarios podrán publicar y editar y borrar sus incidencias.

- **RF-7.** La aplicación dispondrá de una sección de anuncios donde los usuarios podrán publicar y editar y borrar sus anuncios.

- **RF-8.** La aplicación dispondrá de una sección de reuniones donde los usuarios podrán ver todas las reuniones programadas y las reuniones ya realizadas.

- **RF-9.** La aplicación dispondrá de una sección de votaciones donde los usuarios podrán votar.

- **RF-10.** La aplicación dispondrá de una sección de gastos donde los usuarios podrán ver y buscar los gastos realizados en un periodo de tiempo determinado.

- **RF-11**. Los usuarios podrán mandar mensajes a otros vecinos de su comunidad.

- **RF-12.** La aplicación dispondrá de una sección de mensajes donde los usuarios podrán ver sus mensajes enviados y recibidos.

Requisitos Funcionales del presidente:

- **RF-13.** Podrá borrar y editar incidencias y anuncios, aunque no hayan sido publicados por él.

- **RF-14.** Será el encargado de publicar las reuniones indicando la fecha, la hora y el tema que se va a tratar y añadir el resumen de las reuniones que ya han sido realizadas.

- **RF-15.** Será el encargado de publicar las votaciones y el único que podrá ver los resultados.

- **RF-16.** Desde la sección de gastos podrá añadir nuevos gastos que se hayan producido, añadiendo la cantidad y una descripción.

Requisitos No Funcionales:

- **RNF 1.** La aplicación web contará con un Manual de Usuario [ANEXO B].

- **RNF 2.** La aplicación web dispondrá de un filtro de sesión mediante el cual un usuario que no se encuentra logueado solo podrá visualizar la pantalla de inicio, la pantalla de loguin, la pantalla de comprobación de comunidades y las pantallas de registro.

- **RNF 3.** La interfaz gráfica de la aplicación será sencilla e intuitiva haciendo que el uso y la navegación por la web sea apta para todos los usuarios.

- **RNF 4.** La aplicación mostrará mensajes de error si los datos introducidos no son los adecuados.

- **RNF 5.** La primera versión solo estará disponible en castellano.

- **RNF 6.** Se hará uso de ajax para proporcionar dinamismo a la página.

B Manual de Usuario de Community

Figura: Logo de Community

Inicio

Nada más acceder a la aplicación se muestra una pantalla de inicio donde se detallan brevemente las principales funcionalidades que ofrece Community a sus usuarios:

Figura: Pantalla de Inicio

Login

Si pulsamos “Accede al contenido” nos dirigimos a la pantalla de Login:

Figura: Pantalla de Login

Comprobación de la Comunidad y Registro

Si el usuario se encuentra registrado en el sistema podrá loguearse. Si por el contrario todavía no forma parte de este deberá pulsar el botón "Comprobar Comunidad", para comprobar si su comunidad ya está registrada y así poder registrarse como vecino. En la pantalla de "Comprobar Comunidad" se dispone de tres campos de texto que hay que rellenar:

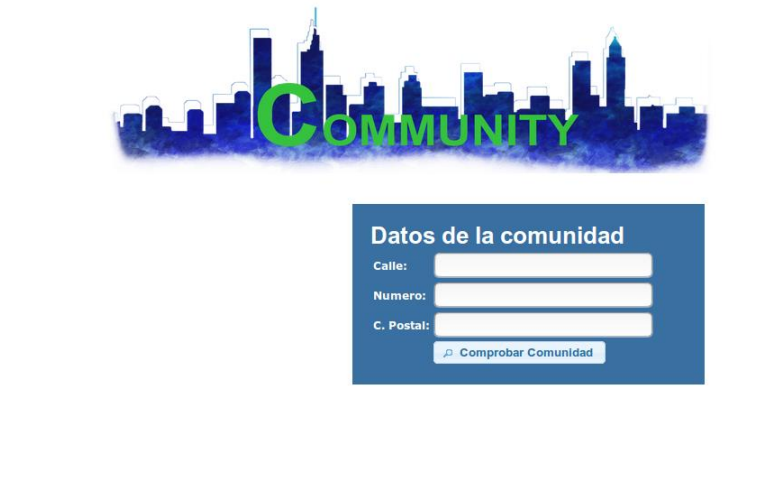


Figura: Pantalla de Comprobación de la Comunidad

Si la Comunidad está registrada nos aparecerá un diálogo con un mensaje indicándolo y un botón que nos redirigirá la pantalla de registro en modo vecino:

Figura: Diálogo indicando que la comunidad está registrada

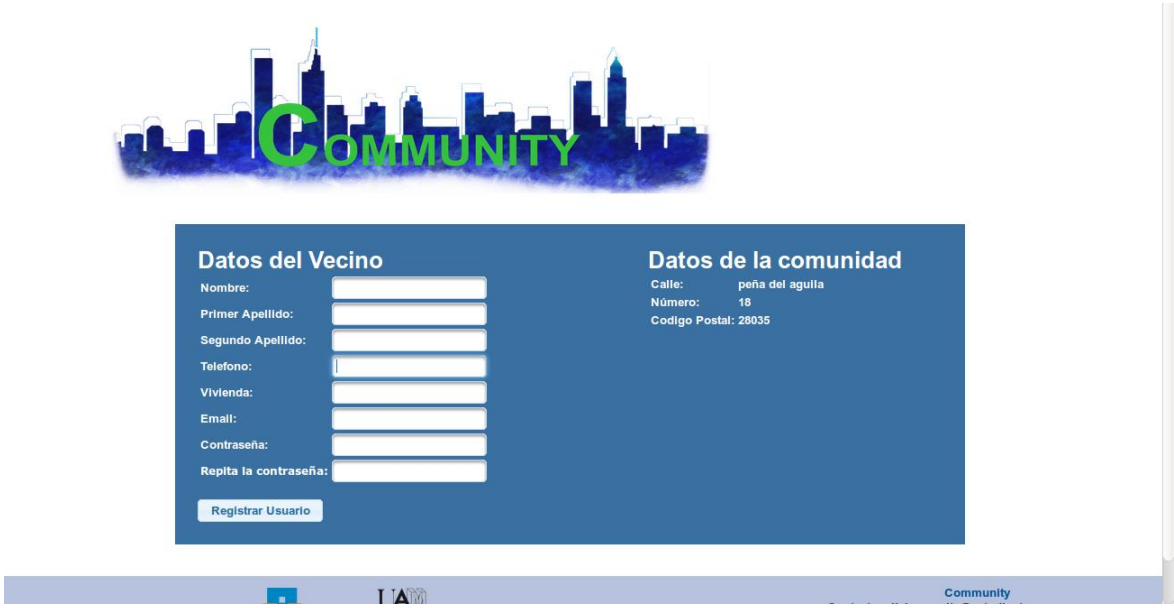


Figura: Pantalla de Registro en modo vecino

Si por el contrario la Comunidad no está podremos registrarnos como presidente virtual de esta:

Figura: Diálogo indicando que la no comunidad está registrada

Figura: Pantalla de Registro en modo presidente

Sección de Inicio (Tutorial)

Una vez hemos accedido al sistema lo primero que veremos será la sección de Inicio. En esta sección se muestra una breve descripción de las funcionalidades de cada sección y como están organizadas:

Figura: Sección de Inicio

En la parte superior derecha de la pantalla aparece el nombre del usuario logueado, un botón que nos permitirá cerrar la sesión y salir de la aplicación, y un mensaje "Tiene mensajes sin leer", que aparece si hay mensajes sin leer en la bandeja de entrada del usuario logueado.

Sección de Usuarios

En la sección de Usuarios se muestra una tabla con todos los usuarios que pertenecen a la comunidad del usuario que se ha logueado:

Figura: Sección de Usuarios

En la tabla se pueden filtrar los usuarios por nombre, apellidos o correo electrónico. Si se pulsa sobre algún usuario de la lista se mostrará un diálogo con la información de contacto del usuario seleccionado:

Figura: Filtro de Usuarios

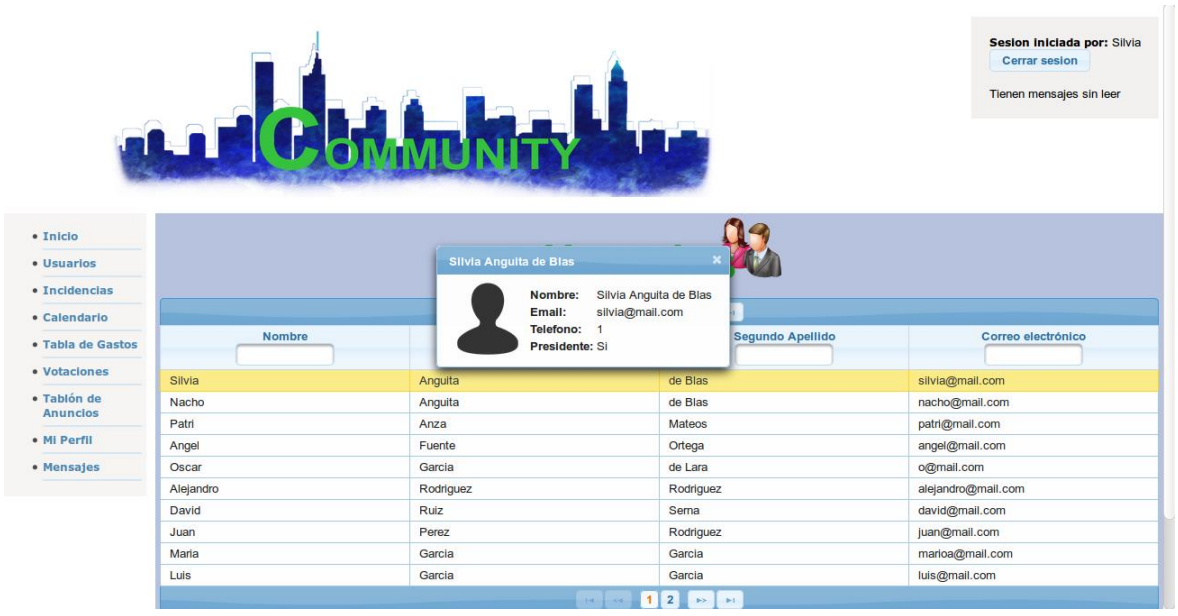


Figura: Diálogo con información del usuario seleccionado

Sección de Incidencias

En la sección de Incidencias se muestra un tablón con tres secciones que dividen las incidencias. En el tablón "Todas" se muestran todas las incidencias, "En Gestión" se encuentran aquellas que todavía no han sido solucionadas y "Cerradas" contiene aquellas que se han solucionado:

Figura: Sección de Incidencias

Las incidencias naranjas son aquellas que se encuentran en trámites de gestión y las azules son las que ya se han solucionado. Pueden añadir incidencias todos los usuarios, pero solo pueden borrarlas y editarlas los autores o el presidente de la comunidad. Las incidencias cerradas no podrán ser ni borradas ni editadas. Para insertar una incidencia hay que rellenar todos los campos y el sistema no permite insertar una incidencia repetida:

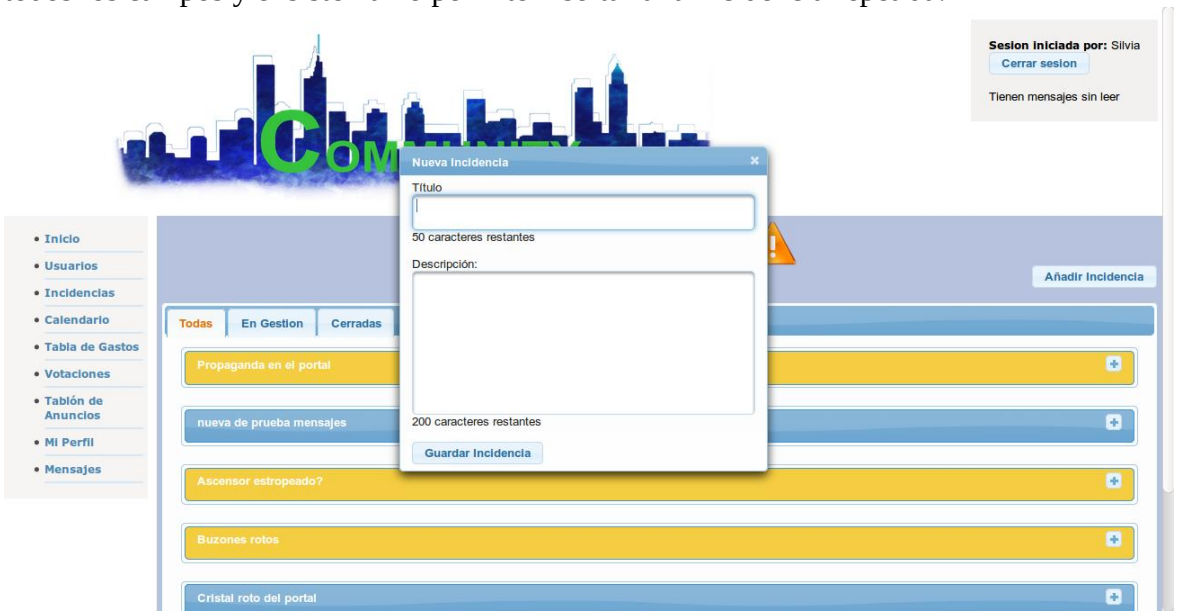


Figura: Diálogo para añadir una nueva incidencia

Sección de Calendario

En la sección de Calendario se muestran los eventos de la comunidad. Solo puede añadir, editar y borrar eventos el presidente de la comunidad. Los vecinos pueden pinchar sobre un evento para ver la información de estos. Los eventos azules son los que ya han pasado y los verdes son los eventos próximos:

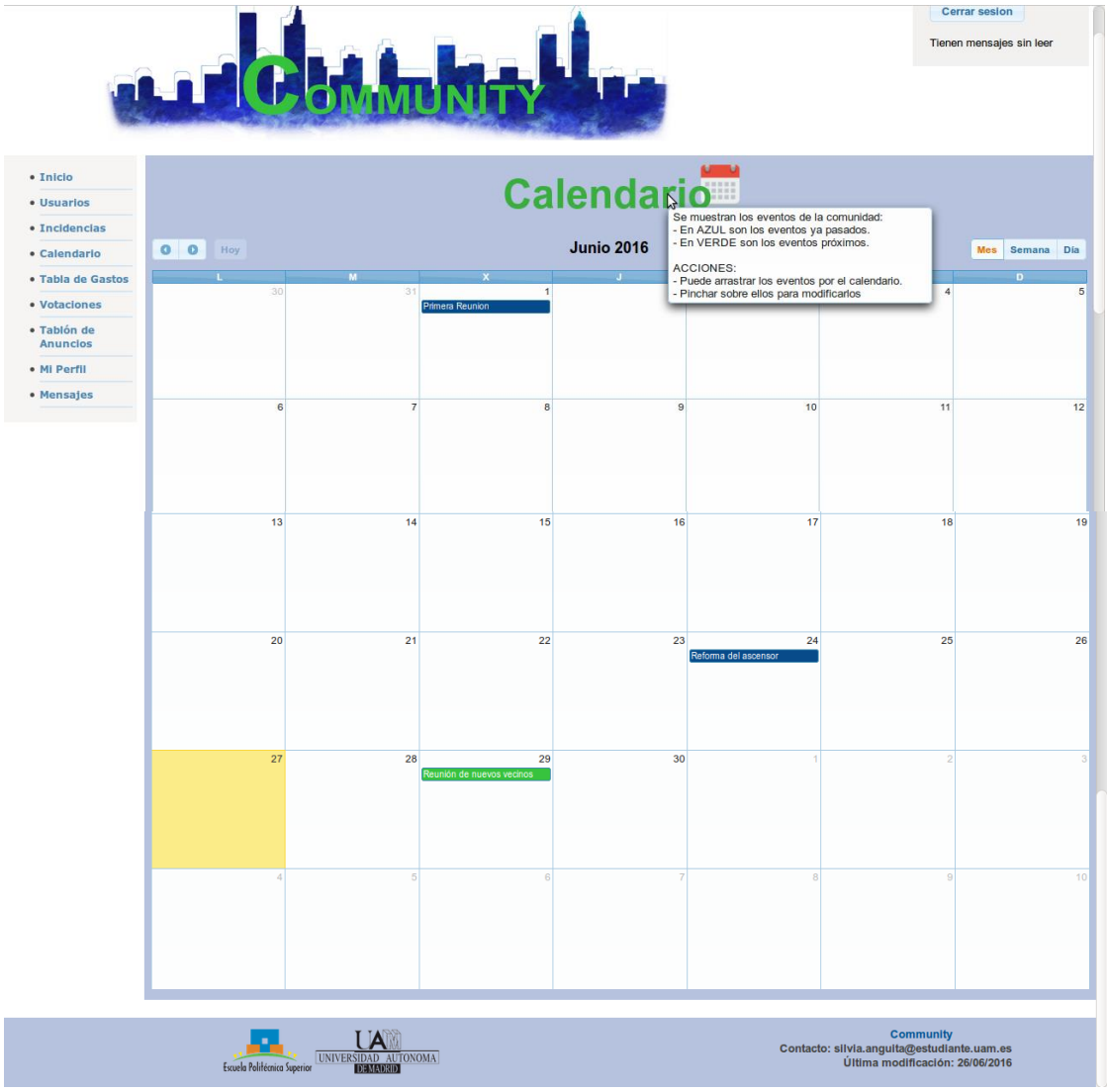


Figura: Sección de Calendario

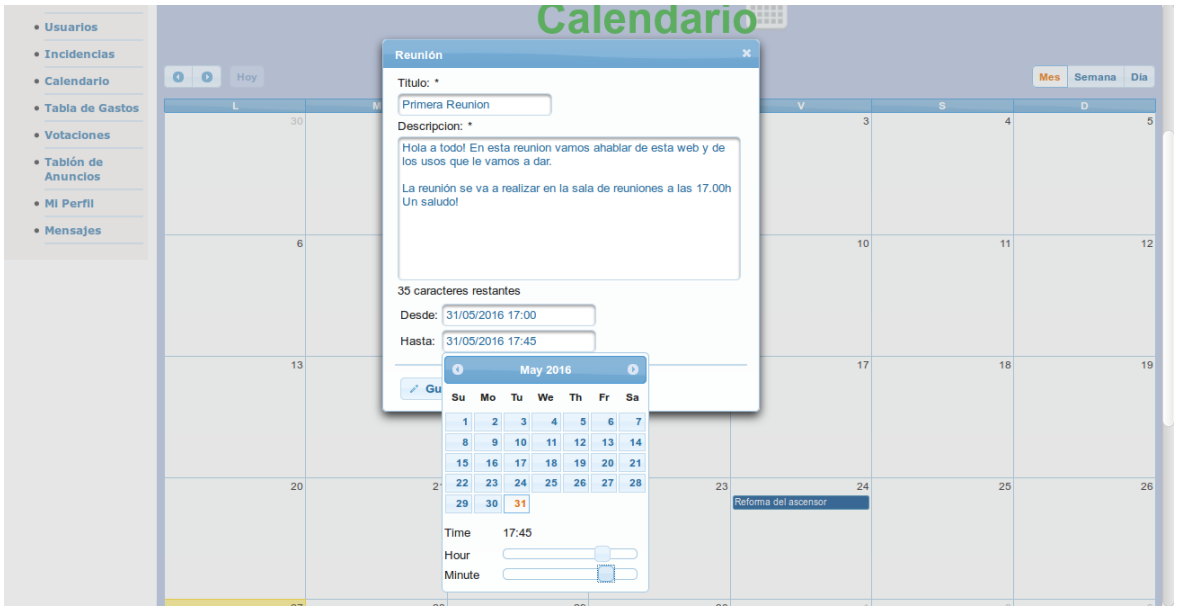


Figura: Diálogo de añadir y editar un evento del calendario

Sección de Tabla de Gastos

En la sección de Tabla de Gastos se muestran los gatos de la comunidad en forma de gráfico de barras y en forma de tabla. El botón de añadir gasto solo le aparece al presidente de la comunidad, debido a que es el único autorizado para añadir gastos, pero debe rellenar todos los datos del diálogo. Si se pincha sobre un gasto de la tabla se abre un diálogo que muestra más información. La tabla se puede exportar pinchando en los iconos, en formato PDF, Excel y CSV. Los datos se pueden filtrar por mes y año:

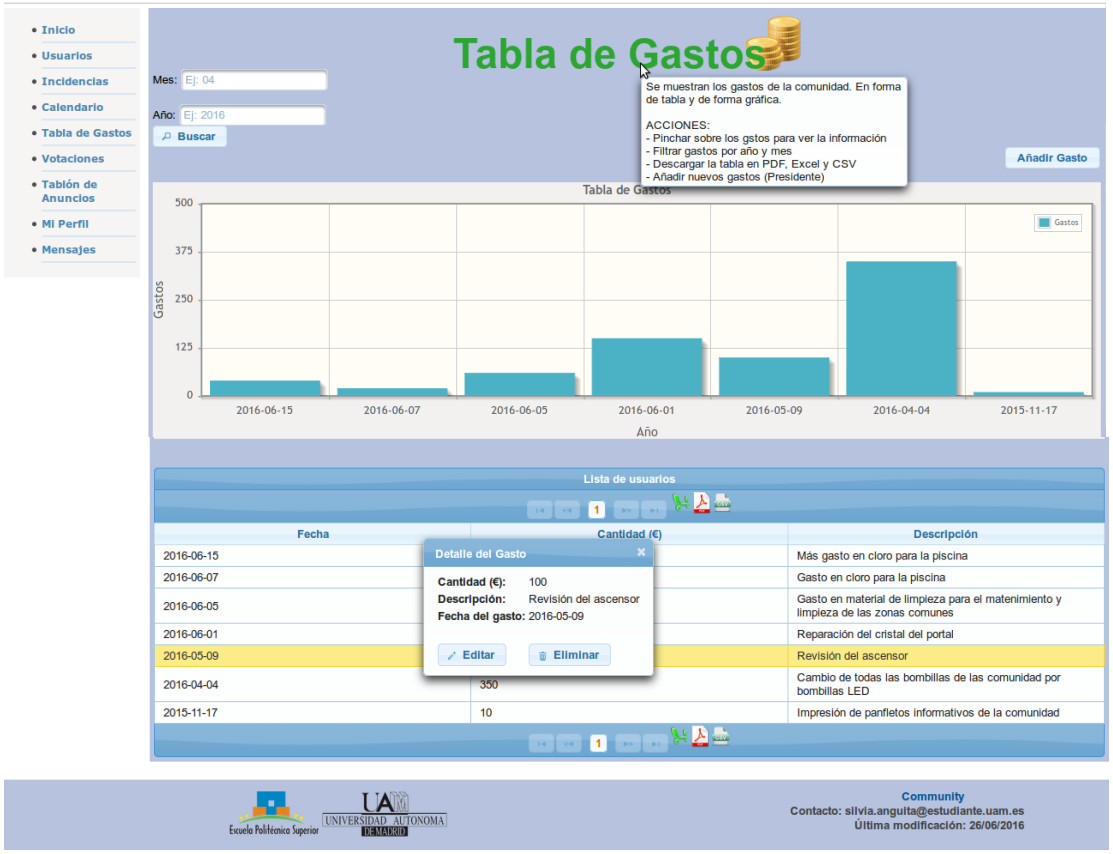


Figura: Sección de Tabla de Gastos

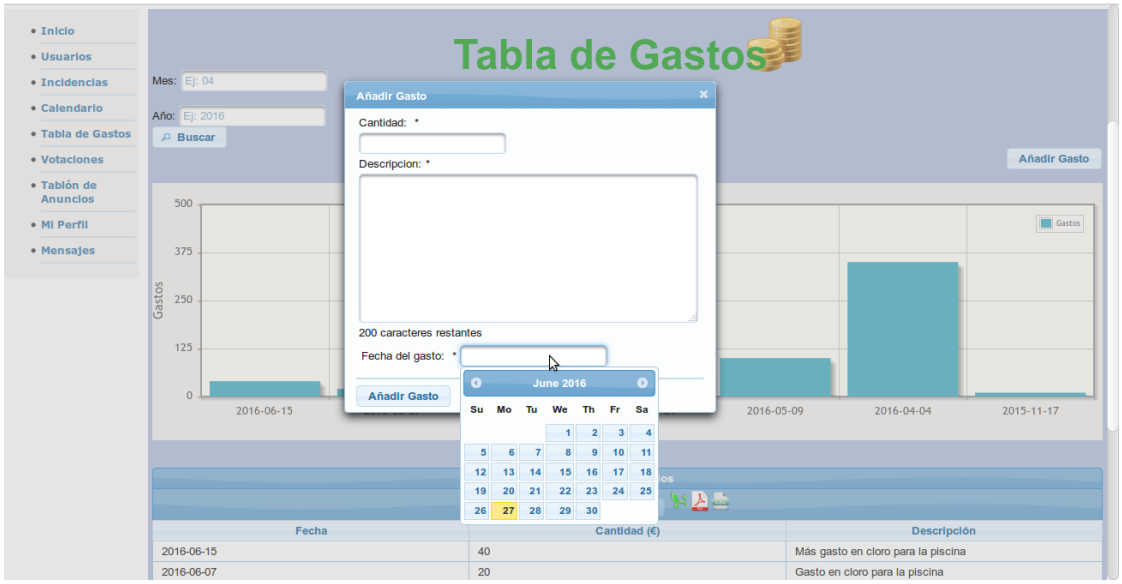


Figura: Diálogo de añadir gasto

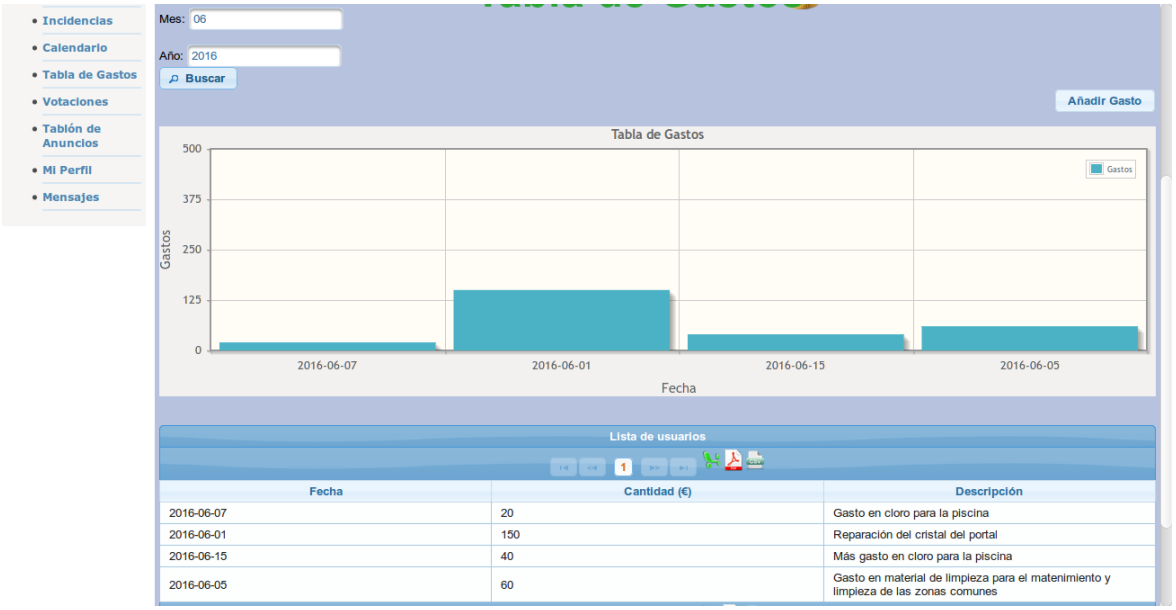


Figura: Gastos filtrados por mes y año

Sección de Votaciones

En la sección de Votaciones se muestra un tablón con cuatro secciones que dividen las votaciones. En el tablón "Todas" se muestran todas las votaciones, "Abiertas" se encuentran aquellas en las que todavía no se ha pasado la fecha límite para contestar, "Cerradas" contiene aquellas en las que ya se ha sobrepasado la fecha límite y "Abiertas y sin Responder" son aquellas que están abiertas y el usuario todavía no ha contestado:

Figura: Sección de Votaciones

Las votaciones naranjas son aquellas que estan abiertas y sin responder, y las azules son las que ya se han respondido o están cerradas. El botón de añadir votación solo le aparece al presidente de la comunidad, debido a que es el único autorizado para añadirlas, pero debe

rellenar todos los datos del diálogo. De este modo, el presidente también es el único autorizado a editar y borrar votaciones, pero estas acciones solo se pueden realizar sobre votaciones abiertas:

Figura: Votación abierta y sin responder

El diálogo de añadir votación cuenta con un campo de texto para la pregunta y un botón para añadir campos de texto para las respuestas. Es obligatorio introducir un mínimo de 2 respuestas y un máximo de 4 respuestas, en caso contrario el sistema mostrará mensajes de error y el diálogo permanecerá abierto:

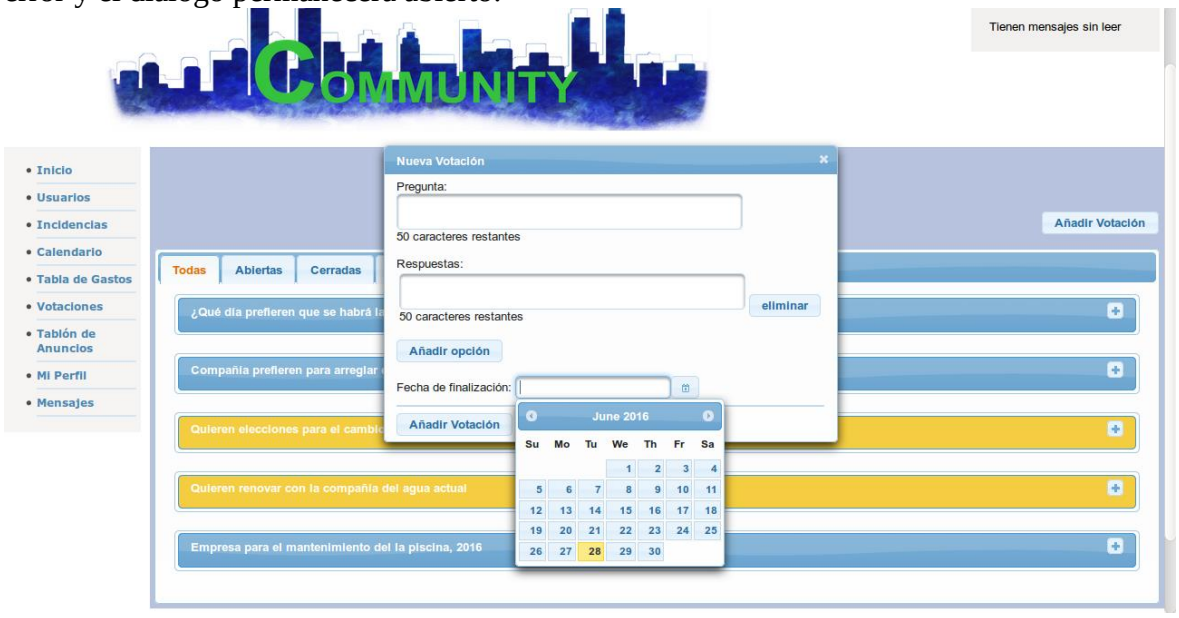


Figura: Diálogo de añadir votación

El presidente es el único que puede ver los resultados de las votaciones, estos se muestran de forma gráfica:

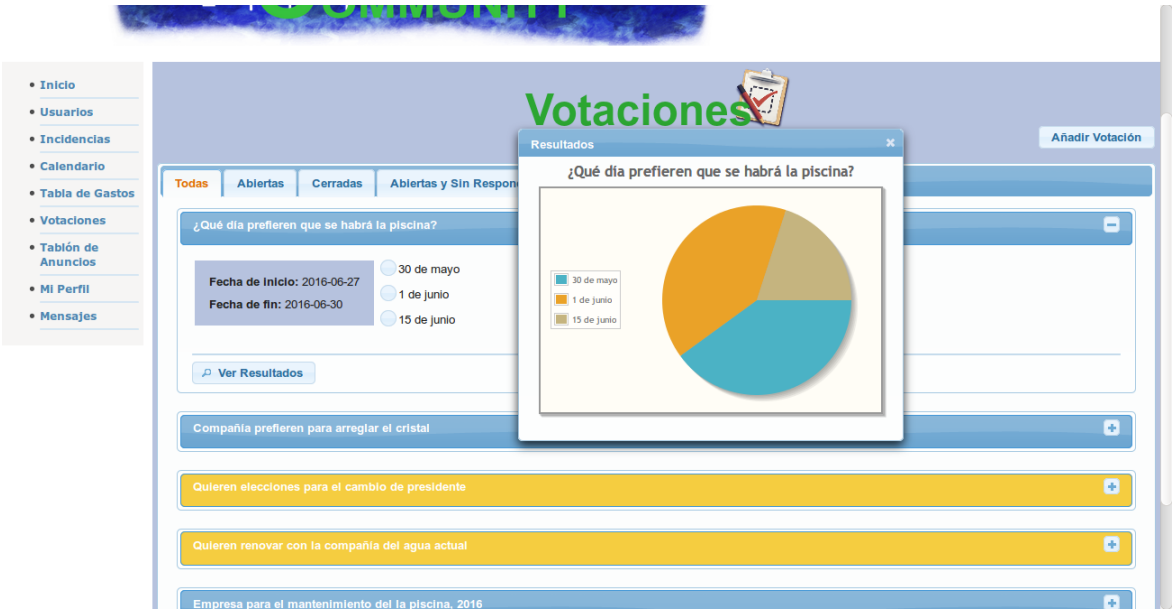


Figura: Diálogo de resultados de una votación

Sección de Anuncios

En la sección de Anuncios se muestra un tablón con tres secciones que dividen las incidencias. En el tablón "Todas" se muestran todos los anuncios, "Abiertos" se encuentran aquellos que todavía no han sido cerrados y "Cerrados" contiene aquellos que se han solucionado:

Figura: Sección de Anuncios

Los anuncios naranjas son aquellos que se encuentran abiertos y las azules son las que ya se han cerrado. Pueden añadir anuncios todos los usuarios, pero solo pueden borrarlos y editarlos los autores o el presidente de la comunidad. Los anuncios cerrados no podrán ser ni borrados ni editados. Para insertar o editar un anuncio hay que rellenar todos los campos:

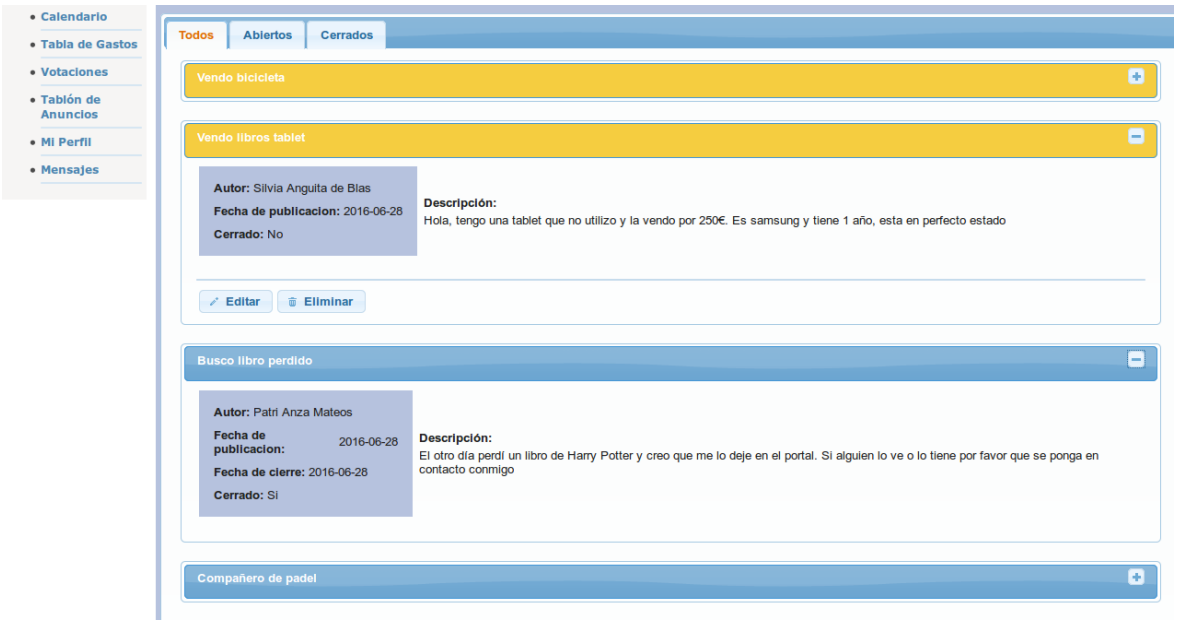


Figura: Diferencias entre anuncios cerrados y abiertos

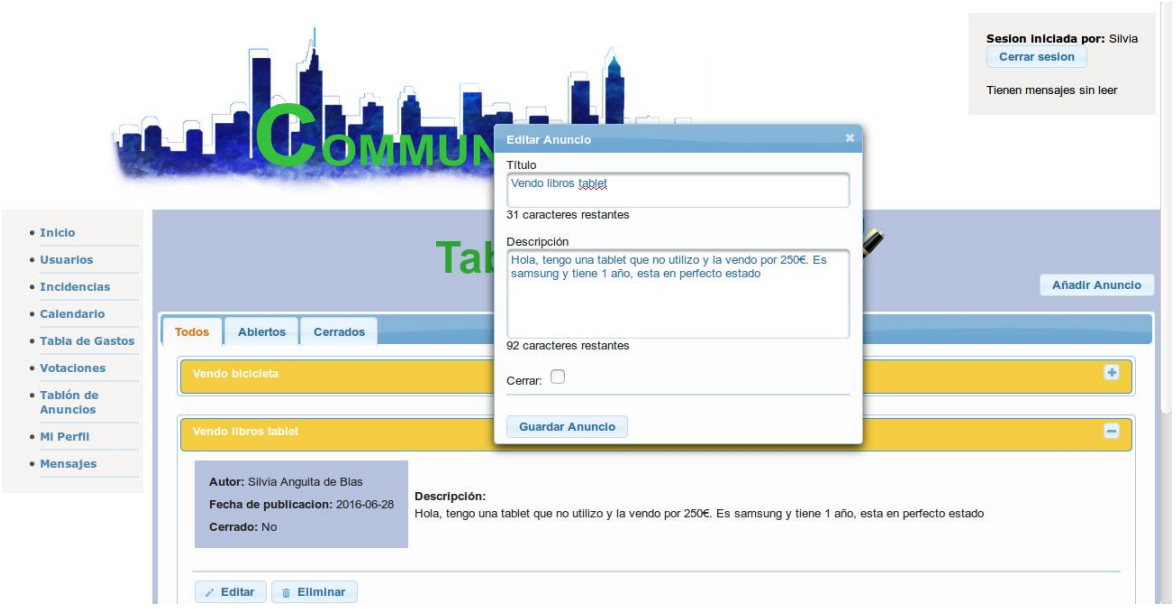


Figura: Diálogo para editar anuncio

Sección de Mi Perfil

En la sección de Mi Perfil se muestran los datos asociados con el usuario logueado y con la comunidad a la cual pertenece. El usuario puede editar sus datos y cambiar su contraseña:

Figura: Sección de Mi Perfil

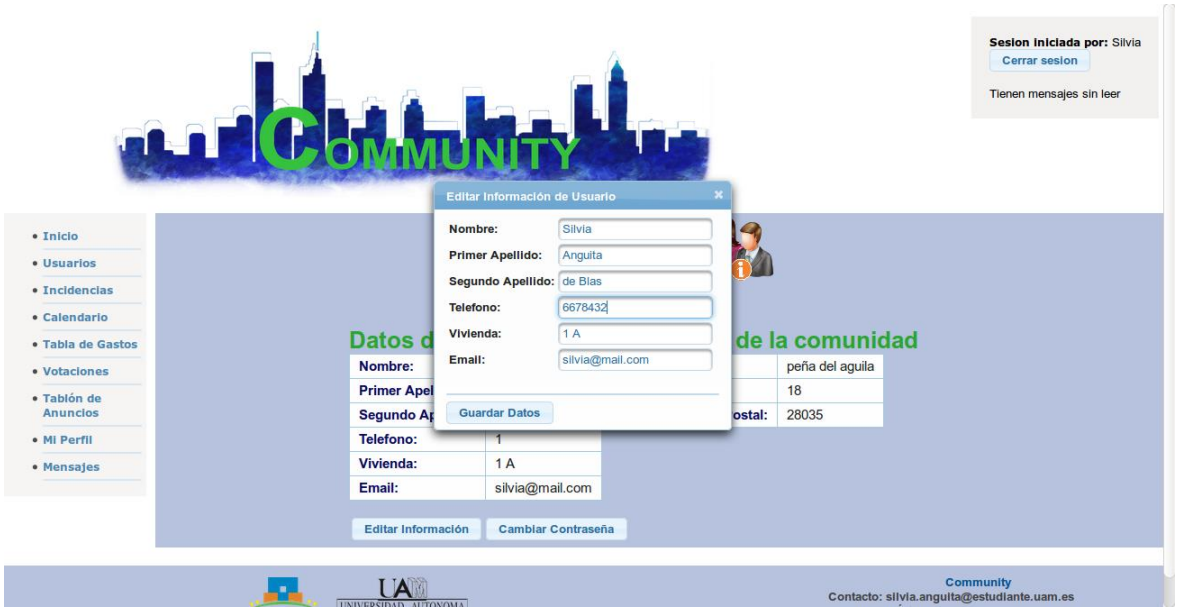


Figura: Diálogo para modificar los datos de usuario

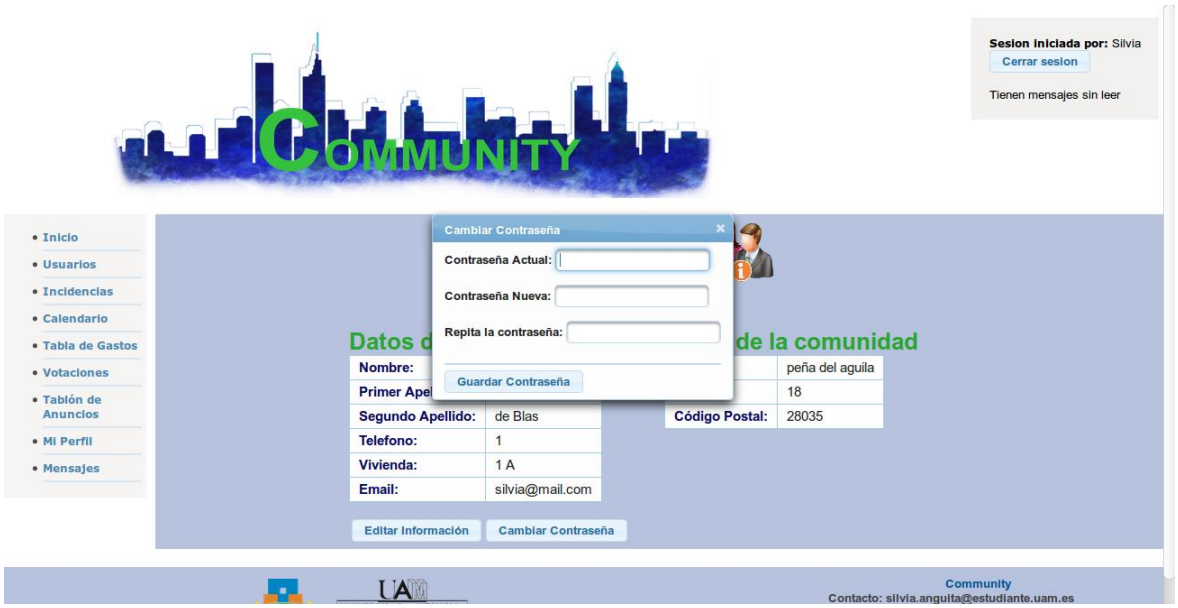


Figura: Diálogo para cambiar la contraseña

Sección de Mensajes

En la sección de Mensajes se muestra un tablón con dos secciones que dividen los mensajes. “Bandeja de Entrada” para los mensajes recibidos y “Bandeja de Salida” para los mensajes enviados. La bandeja de entrada muestra los mensajes en dos colores distintos. Los mensajes de color blanco son lo que ya han sido leídos y los mensajes de color oscuro son los que todavía no han sido abiertos. Si no quedan mensajes por leer, el texto “Tiene mensajes sin leer” que aparece en el cuadro de sesión desaparece:

Figura: Sección de Mensajes

Además, el sistema manda mensajes automáticos cada vez que se realiza un registro en alguna de las otras secciones:

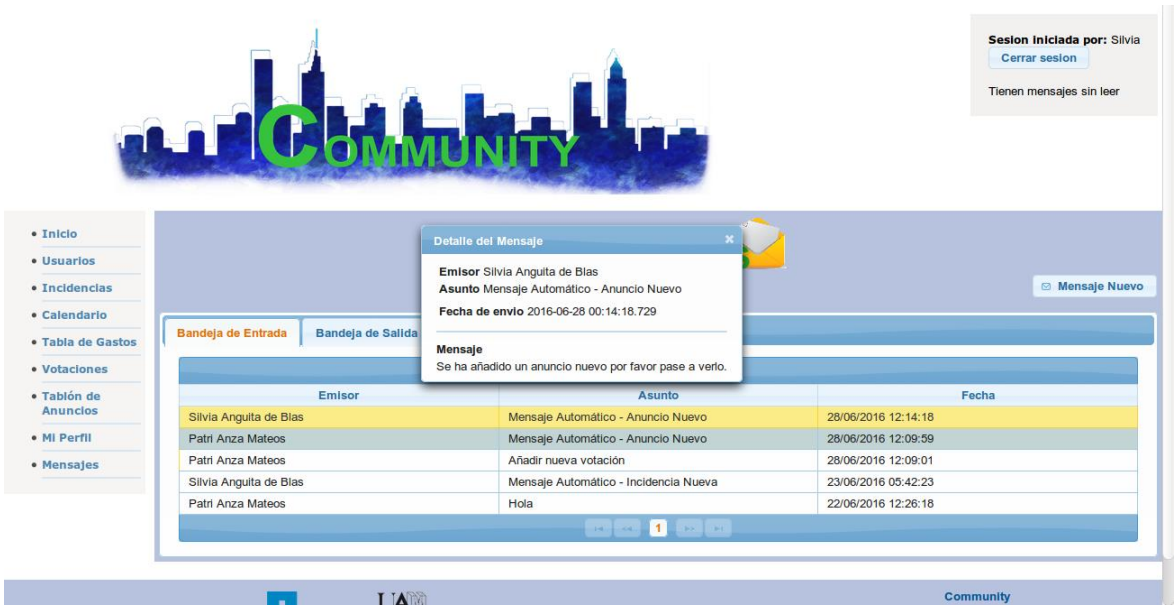


Figura: Detalle de un mensaje automático

Se pueden enviar mensajes a cualquier usuario que pertenezca a la misma comunidad. Para poder enviar el mensaje hay que rellenar todos los campos del diálogo:

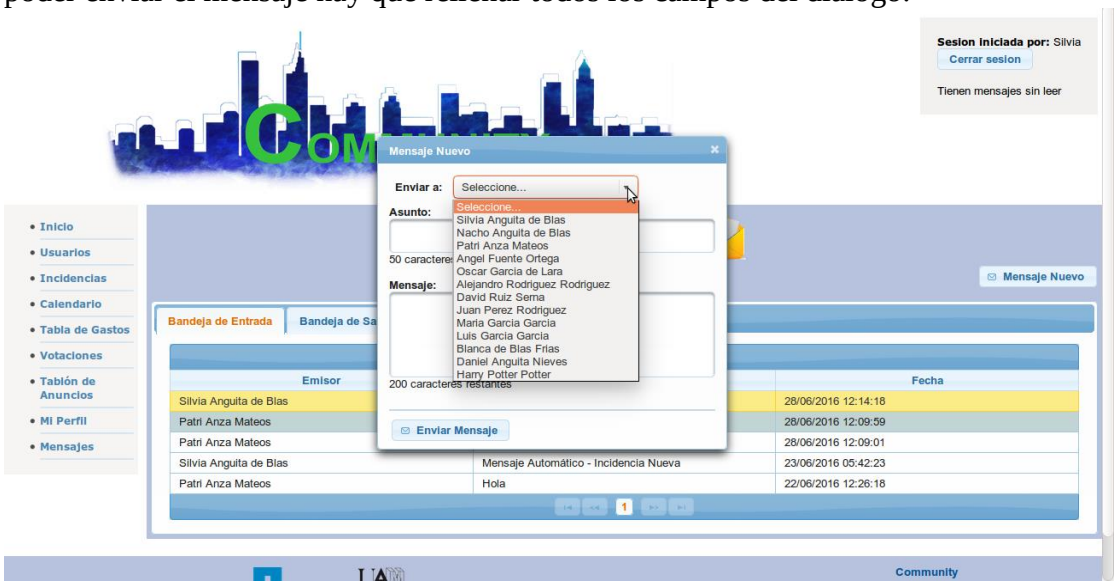


Figura: Diálogo para enviar mensaje